Measuring similarity of classified advertisements using images and text

with applications to recommendation and search at *finn.no*

Audun Mathias Øygard

Thesis
for the degree of
Master of Science
Modelling and Data Analysis

Faculty of Mathematics and Natural Sciences
University of Oslo
September 2015

Thesis advisor: Gudmund Hermansen & Nils Lid Hjort Audun Mathias Øygard

Measuring similarity of classified advertisements using images and text

Abstract

A basic problem in large collections of documents is to find similar items, for basic search, recommendation, or other purposes. Traditionally techniques for this have relied only on the text in the document, since analysing images has been difficult and costly. However, recently, new methods for analysis of images via deep convolutional neural networks have made it possible to efficiently classify and compare images, which opens up for using images as additional information when comparing documents. In our case we will explore methods of comparing classified ads in the Norwegian online marketplace *finn.no*, utilizing both images and text in the ads, so called multi-modal comparison. We will evaluate two methods using both images and text for comparing classified ads, one based on representing images and text with features from respectively a deep convolutional neural network and a topic model, and one based on combining classifier output via a separately learned word-embedding model that retains "semantic" similarity between words. We will compare these two methods with baseline comparison methods utilizing only text or only images.

Contents

Dedicated to Astrid and Eskil.

Acknowledgments

Many thanks to *finn.no* for agreeing to supervise me and provide me with the ad data used in this thesis. Also thanks to my supervisor Nils for giving me the opportunity to write this thesis, and assisting supervisor Gudmund for many interesting and fruitful conversations.

1 Introduction

A BASIC PROBLEM IN LARGE COLLECTIONS OF DOCUMENTS is to find similar items, for basic search, recommendation, or other purposes. Traditionally techniques for this have relied only on the text in the document, since analysing images has been difficult and costly. However, recently, new methods for analysis of images have made it possible to efficiently classify and compare images, which opens up for using images as additional information when comparing documents. In our case we will explore methods of comparing classified ads in the Norwegian online marketplace *finn.no*, utilizing both images and text in the ads, so called multi-modal comparison. We will evaluate two methods using both images and text for comparing classified ads, one based on representing images and text with features from respectively a deep convolutional neural network and a topic model, and one based on combining classifier output via a separately learned word-embedding model that retains "semantic" similarity between words. We will compare these two methods with baseline comparison methods utilizing only text or only images.

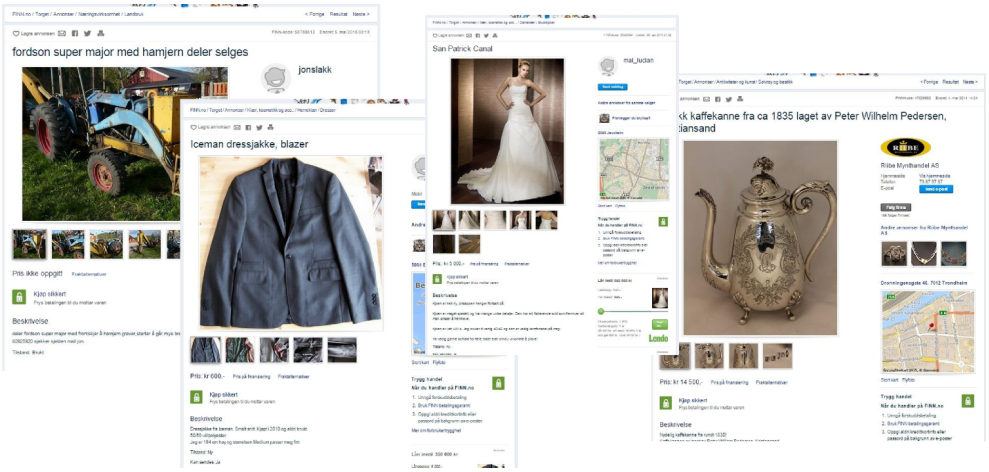


Figure 1.1.1: Examples of online classified ads from *finn.no*

1.1 Description

Finn.no is an online classified ads marketplace, where people can publish classified ads for objects they want to sell or give away. It is the largest classified ad marketplace in Norway, and at any time there is more than 300 000 classified ads for objects ranging from stamp collections to industrial machinery.

In such a large collection, finding and retrieving relevant ads is central to the perceived quality of the marketplace. Usually comparison/retrieval of classified ads will be done solely based on text content, but often a complicating matter is that the quality of text descriptions in the classified ads ranges from excellent to extremely terse, see e.g. figure 1.1.2.

In these cases it would be very helpful to be able to use information from the images in the classified ads. Recently, new methods for training deep convolutional neural networks have enabled state-of-the-art performance on very varied image classification tasks, see e.g. Krizhevsky et al. (2012). We would like to use these methods to be able to compare the classified ads using information from both text and images. More precisely, we want to learn an embedding function $f(txt, im) : R^n \rightarrow R^m$, so that we can measure similarity of ads via some distance metric, e.g. the Euclidean norm:

$$\text{sim}(x, y) = ||f(txt_x, im_x) - f(txt_y, im_y)|| \tag{1.1}$$

Figure 1.1.2: Example of erroneously categorized ad lacking descriptive text. The ad is for a pair of skis, but is categorized as silverware, presumably due to erroneous user input.

Applications for a similarity measure as above can be to e.g. find similar ads to ones we are interested in, improvements to text-search (more relevant results), corrections of errors in ads, recommendations of relevant ads, or creation of new categories via clustering.

1.2 Data

Our dataset consists of 3.21 million classified ads from *finn.no* from 2014. Typically the ads consist of a header text, a brief text description of the object and one or more images of the object, see figure 1.1.1. The ads in our dataset contain around 12 million images, i.e. on average 3.7 images per ad, but this varies widely for the type of object being sold. The marketplace additionally has a hierarchy tree of categories, with the user assigning each ad to one of the "leaf" categories in the tree. Altogether *finn.no* has around 500 such categories, for instance "Dyr og utstyr/Hester/Ponnier" ("Animals and equipment/Horses/Ponies") or "Næringsvirksomhet/Verksted, bygg og anlegg/Treforedling" ("Business/Workshop, construction/Wood processing"). Many categories are

mixed categories, e.g. "Elektronikk og hvitevarer/Annet" ("Electronics and appliances/Other"), containing items which do not clearly belong to any other subcategories. For our task we removed all these categories and some other generic categories, to get a subset of only 270 categories. See appendix 7.1 for the exact categories we used.

A significant problem with the classified ad dataset is that users are free to assign ads to any category they please, and there is generally very little error correction from *finn.no*'s side. Because of this there are a lot of user-caused errors in the data. Many of the classified ads are put in the wrong category, and many classified ads contain several items that belong to different categories. Based on a small sample of ads we estimated that the amount of ads that are wrongly classified are approximately 5%, and the amount that contain several items of multiple classes are around 6%. Since cleaning up all these errors would be a very time-consuming task we've used the ads as they are with minimal corrections. We keep this in mind when evaluating the precision of our classifiers later on.

1.3 Our approach

Our approach to create a similarity measure utilizing both images and text, is to first train a convolutional neural network classifier (see chapter 3.2) which we can either extract image representations from or use as a regular classifier. Additionally we train a topic model (see chapter 4.2) which we can use to create representations of the text of our ads, and an ad classifier based on these topic representations. We then evaluate two approaches for combining the image and text information in a similarity measure via an embedding (as in (1.1)) : one based on using the learned image and text "features" directly as embeddings, and a second method based on projecting the probability output of our learned classifiers into "semantic" word-embeddings which are separately learned on a large body of text. These word-embeddings, which we will describe in chapter 5.2.1, manage to encode "semantic" information about similarity between objects of different types, and thus may be useful for measuring similarity between items in *finn.no*. Both these learned embeddings can then be used with a distance metric as in (1.1) to create a similarity measure.

To evaluate the quality of our similarity measures, we will create multiple test

sets of ad triplets, (x, x^+, x^-), where the task is to correctly rank which of ads x^+ and x^- are most similar to ad x. This way of scoring our similarity measures is a somewhat crude method, but makes it significantly simpler to generate evaluation sets from our data. For more details on how we construct these sets, see chapter 2.3.

Note that our approach is based mostly on discriminative models, i.e. we do not try to jointly model class labels and text and images as would be done in generative models. The reason for this is twofold: generative modeling of images is very difficult, and discriminative modeling tends to be more precise than generative modeling in cases where we have large amounts of data. The models we learn are at least partially "black-box" models, but since we're not explicitly interested in inspecting the internals of our models, this is a minor nuisance. We also note that projecting the class probabilities into a semantic vector-space doesn't make much sense from a probabilistic perspective, but our goal is a discriminative task (to correctly rank similar ads) and thus the probabilistic interpretability of our model is not something we are inherently interested in.

1.4 Implementation

There does not exist ready-made software packages for comparing documents using the methods we describe, therefore a considerable amount of time was spent on implementing them. As in many data analysis tasks, much time was also spent on cleaning and formatting the data, and finding suitable parameters for our models via cross-validation, see e.g. chapter 4.3.2.

Training the image classifier we describe also required computers equipped with high-performance graphical processing units in order to get results in a timely manner. We opted to lease computational capacity from the cloud service *Amazon EC2*[1], since we only need it during training, and it thus would not be worthwhile to e.g. purchase this hardware. Thus some time was also spent setting up and configuring the cloud service.

[1] https://aws.amazon.com/ec2/

1.5 Similar work

There are a number of methods for comparing images via a similarity metric, but we'll only mention some recent methods that are related to our approach. Norouzi et al. (2013) is the main inspiration for our methodology, and uses a *deep convolutional neural network* classifier that projects class probabilities into a separately learned semantic embedding space where we can use an arbitrary distance metric. Chechik et al. (2010) uses a deep convolutional network classifier similar to the above but independently learns a linear embedding of features from the classifier to reduce pairwise ranking loss. Wang et al. (2014) meanwhile learn an similarity metric for images via applying triplet pairwise ranking loss as an objective for a deep convolutional neural network. Wan et al. (2014) has a good overview of these and similar methods for image similarity metrics, and compares the performance of the methods on a couple of data sets.

There are also a number of methods that try to learn joint image and text embeddings in some manner. Kiela and Bottou (2014) use a simplistic method of concatenating features from a deep convolutional neural network classifier and the word vectors from a semantic word embedding in order to compare images and text via Euclidean distance between the concatenated feature vectors.

1.6 Layout of thesis

In the second chapter we give a brief introduction to *similarity learning*, introduce some terminology we will find useful, and describe how we may evaluate the similarity measures we create. In chapter three, we describe the theory and practice behind the method we use for image analysis and classification, *deep convolutional neural networks*, as well as evaluate methods which can be used for calculating image similarity via these models. We note that a large portion of the thesis is spent in this chapter as the techniques for image analysis may be novel to many. The fourth chapter describes how we may create features from text via a *Latent Dirichlet Allocation* topic model, and how we train a logistic regression classifier based on these features. We briefly evaluate the resulting accuracy of the classifier as well as a similarity measure based on the learned features. The fifth chapter describes how we can combine the information from the text and images to produce a

"multi-modal" similarity measure. A "semantic" word-embedding model is briefly described, as well as how it's trained. The final chapter is spent summarizing our results, discussing how the methods may be applied in *finn.no*, and suggesting some possible improvements.

2

Similarity learning

Our task is to learn a way to compare classified ads from *finn.no*. In this chapter we will give a brief introduction to the general problem of *similarity learning*, introduce some terminology we will use, and describe how we may evaluate the similarity measures we create.

2.1 Similarity learning

Measuring similarity between items is a central task in many applications, such as information retrieval, clustering or recommender systems. Commonly one wants to identify items that are similar to each other in some sense in a large dataset, in order to e.g. find the most relevant items (information retrieval), recommend similar items (recommender systems) or find groups of similar items (clustering).

Such similarity measurement requires a way to represent items and a way to measure the similarity between these representations. Manually finding the best way to represent and measure similarity may be a very time-consuming task, and it

is therefore attractive to learn such representations and measures in a supervised manner from data. This problem is in general referred to as *similarity learning*[1], and we will give a brief definition of two such common settings below, that of *regression similarity learning* and *pairwise similarity learning*.

In the *regression similarity learning* setting, we have a set $\mathcal{S}$ of pairs of items (x_1, x_2) and a measure of similarity between the items $y \in \mathbb{R}$, and we wish to estimate a similarity measure $\text{sim}(x_1, x_2)$ so that

$$\text{sim}(x_1, x_2) \approx y \quad \forall \quad (x_1, x_2, y) \in \mathcal{S} \tag{2.1}$$

This is an estimation problem, and can thus in many cases be solved via ordinary regression methods from statistics.

However, in many similarity learning settings, ground truth data on measures of similarity y between pairs of items (x_1, x_2) is not easily available, or may be costly to acquire. Instead, we may use available data on which items are more similar than others, i.e. information that item x is more similar to item x^+ than x^-. This is referred to as *pairwise similarity learning*, and we in this case seek to learn $\text{sim}(x, y)$ so that

$$\text{sim}(x, x^+) < \text{sim}(x, x^-) \quad \forall \quad (x, x^+, x^-) \in \mathcal{S} \tag{2.2}$$

This is a weaker restriction than in (2.1) above, but if we have sufficient examples available, it is hoped that this will give similar results as in the regression similarity learning case.

A more thorough overview of similarity learning can be found in e.g. Kulis (2012) or Bellet et al. (2013). We should mention that the field of similarity learning is mostly focused on general methods for learning similarity measures directly from the data and not task-specific representations such as the image or text representations we will use[2], but we still find it useful to think of our problem in terms of the similarity learning framework. We can also mention that similarity learning is closely related to the problem of *learning to rank* (for an introduction to this problem, see e.g. Hang (2011)), and in many cases similarity learning problems can be cast as learning to rank problems. The field of *learning to rank* is however generally more interested in correctly ranking relevant items based on e.g.

[1] or also distance/metric learning

[2] though nonlinear image representations similar to the ones we introduce in chapter 3 is mentioned in Kulis (2012, Chapter 3.2.3)

text-based queries, and not inherently interested in learning a similarity measures, though similarity measures are often used for the task.

2.2 Some terminology

We will formulate the problem to learn a similarity measure $\text{sim}(x, y)$ fulfilling the requirements in (2.2) as a more specific task. We seek to learn a vector representation $f(x) \in \mathbb{R}^n$ of the classified ads, based on the text and/or image content of the ads. We call this an *embedding*, and together with a given distance metric $d(x, y)$, we define our similarity measure as:

$$\text{sim}(x, y) = d(f(x), f(y))$$

Thus our focus is on learning a proper embedding, and we take the distance metrics for granted. We will describe in the following chapters how to learn such embeddings for respectively text, images, and combinations of images and text. For our similarity measure we will evaluate two commonly used distance metrics, the Euclidean distance:

$$d(x, y) = \sqrt{\sum_{i=1}^{n} (x_i - y_i)^2} \tag{2.3}$$

as well as the cosine distance:

$$d(x, y) = 1 - \frac{\sum_{i=1}^{n} x_i \cdot y_i}{\sqrt{\sum_{i=1}^{n} x_i^2} \cdot \sqrt{\sum_{i=1}^{n} y_i^2}} \tag{2.4}$$

The cosine distance is based on the cosine angle between two vectors $cos(\theta) = \frac{x \cdot y}{||x|| \cdot ||y||}$, thus we can roughly say that cosine distance focuses on the *angle* between vectors, while the Euclidean distance focuses on the *magnitude* of vectors. The cosine distance is especially popular in computational applications, since only entries where both vectors are non-zero need to be evaluated, and thus it can be more efficient to calculate than the Euclidean distance. We note that the cosine distance is not a proper distance metric, as it does not fulfill the triangle inequality.

2.3 Evaluation

To evaluate our similarity measure, we need some way of scoring it. We do not have available ground truth data as in regression similarity learning, and creating such a set would be extremely time-consuming and difficult, since we would need to annotate it ourself, and also since similarity between ad objects may be ambiguous and hard to precisely specify. Instead we resort to a scheme for creating triplet examples, (x, x^+, x^-), as in pairwise similarity learning. We can thus score our similarity measure by how large share of these triplets it ranks correctly.

We create three such sets of triplets, two for comparing coarse-grained similarity, and one for fine-grained similarity. The first two sets can be based on the categories of the ads, where we simply sample x and x^+ from the same category, and x^- from a different category. Thus x should always be more similar to x^+ than x^-. The first of these two sets, which we call *coarsest* is based on selecting x^- from any random category, so that if our sample x is from the *t-shirt* category, the negative sample may be from e.g. the *sofa* category. The second of these two sets, which we call *coarse* is based on selecting x^- from a category that is within the same "branch" in the hierarchy of categories as defined by *finn.no*. Thus, if our sample x is from the *t-shirt* category, the negative sample x^- will be from another clothing category, for instance *shirts*. Thus we assume that triplets from the set *coarse* may be harder to correctly rank than triplets from the set *coarsest*. The final set of triplets, which we call *fine*, is based on splitting our ads, in order to generate "new" ads we know should be similar. We take any ad with a sufficient amount of text and more than one image and generate two "new" ads that randomly sample words and images (without replacement) from the original ad. We can then set the two new ads as x and x^+, and set x^- as a similarly created, but different, ad from the same category. This method should allow us to evaluate more fine-grained similarity tasks, as we're comparing ads from within the same category. Note that since we've effectively thrown out half the information for each ad, the scoring may be worse than if we'd used our methods on true similar ads, but it still allows us to compare the methods in a meaningful way.

We sample our triplets from a separate test set of classified ads that were collected from november 2014 to december 2014. Altogether we have approximately 235 000 classified ads in our test set, and from these we generate 10

ooo triplets for each of the sets described above. We note that there may be erroneous categorization of ads in our testset due to wrong user input as described earlier, but since we're most interested in comparing the relative performance of our similarity measures, this won't impact our scoring method too much.

3

Image analysis

The images in classified ads from *finn.no* are a very diverse set of images, and the quality of the images range from very clear images with little or no clutter, to poor images with several elements and obstructions, and even images of completely unrelated items. Given the large variety, it is a challenging task to recognize and correctly classify the items in each image. In this chapter we will draft a method to precisely classify the images, using state-of-the-art image classification methods. The classifier, and features extracted from the classifier, will later be used as part of our multi-modal ad similarity metric.

When analysing images from diverse and complex real settings, such as the images seen in classified ads, training a regular classifier directly on the raw pixels is unlikely to work well except in trivial cases. The reason for this is that the pixels individually carry little information about the overall motive, that the dimensionality of data given by raw pixels in a $N \times M$ image is larger than most classifiers will handle nicely, and that classifiers not based on correlation between different pixels will usually respond poorly to variations in image settings such as

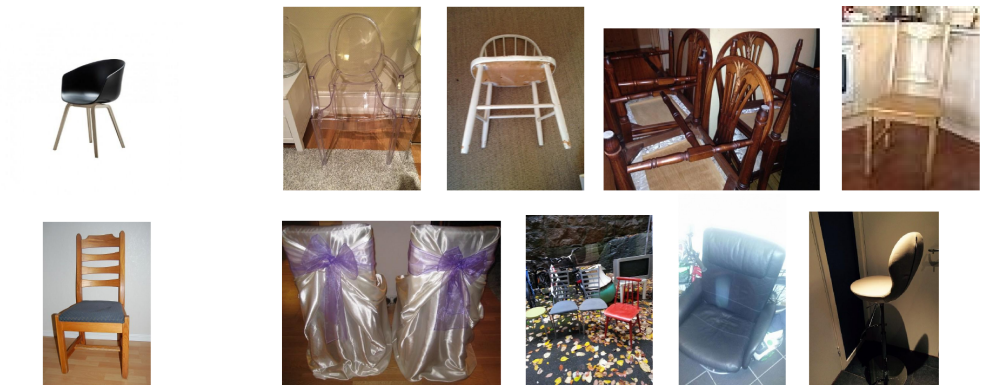


Figure 3.0.1: Examples of "easy" and "challenging" images. All are from the real ads in the category "Stoler og krakker", i.e. "Chairs" in *finn.no*

lighting, scale and angle.

Instead, most methods for image analysis will try to capture more general elements in the image, such as e.g. corners, edges and more complex shapes, via either learned or manually created *features*. There exists a wide array of methods for building such features. Up until recently (2012), the most common of these feature training methods were SIFT (Scale-invariant feature transforms, Lowe (1999)) and HoG (Histogram of Oriented Gradients, Dalal and Triggs (2005))[1]. A drawback of these methods are that they usually require manual training, i.e. feature engineering. However, recently, a more powerful method, called *Deep Learning* has become very popular and shown state-of-the-art performance on several well known image classification benchmarks. In the next part, we will explain the background and current theory for *Deep Learning* methods from the perspective of neural networks.

3.1 "Deep Learning" for feature extraction and classification

"Deep Learning" is a pretty loosely defined term, but generally can be said to refer to any models using layered hierarchical feature representations, where the feature representations are learned from the data. The term "deep" stems from the number of layers in the hierarchy, with any "deep" learning model having more than one

[1] see Malisiewicz (2015) for a good overview of feature learning methods prior to 2012

layer [2]. The term was coined by academics working with neural networks, so the term "deep learning" tends to be conflated with layered neural networks, but there has been a few deep learning methods which doesn't use neural nets at all, such as PCAnet (Chan et al., 2014), multilayered kernel machines (Cho and Saul, 2009) or deep coding networks(Chalasani and Principe, 2013).

The general layout of a deep learning classifier is that several feature transformations are chained together, with the first layer extracting low-level features, typically edges, and outputting it to the second layer. The second layer similarly will use these features to extract mid-level features, say for instance corners or triangles, and output it to the next layer, and so on, increasing the abstraction for each layer. The final layer typically outputs its features to a classifier of some type.

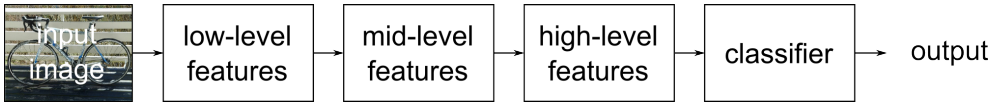


Figure 3.1.1: Example architecture of a deep learning image classifier

3.1.1 An intuitive motivation of deep learning

To motivate the use of such multi-level features for the task of image classification, we will discuss how we might heuristically build a classifier for a single class of visual objects. This is only for illustration, so we don't describe the exact details of how it might work. Let's say we want to be able to classify any images containing a dog. The simplest solution to this problem might be to use some kind of template, e.g. linear filters, which we use to classify the image as containing a dog if gets a high score on our image. Let's say we have a template matching the pose of a dog, see figure 3.1.2.

An obvious problem with this solution is that a dog might take many poses, and so our template will only match one of these poses perfectly and probably match poorly many other poses. A naive extension of this solution would be to train multiple templates, each matching a different potential pose, in hope that this would match most images containing dogs.

[2] but the trend seems to be to add more and more layers, with the winning ILSVRC 2014 model, GoogLeNet, using 22 layers

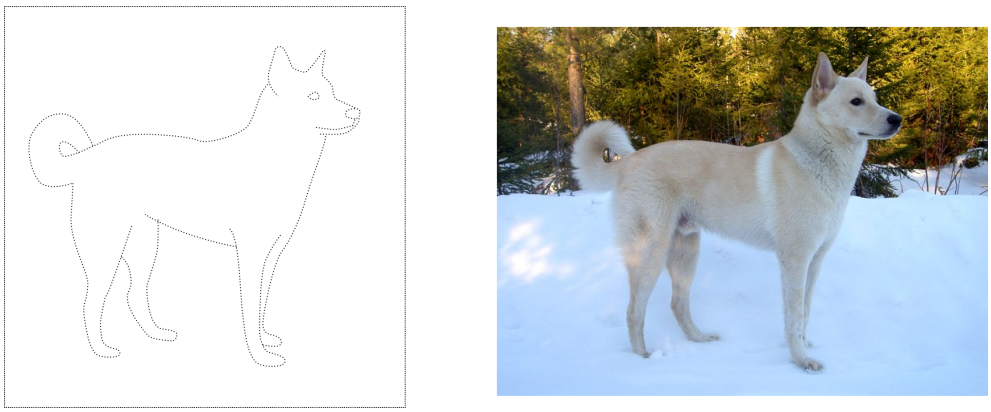


Figure 3.1.2: A sketch of a template to detect a dog

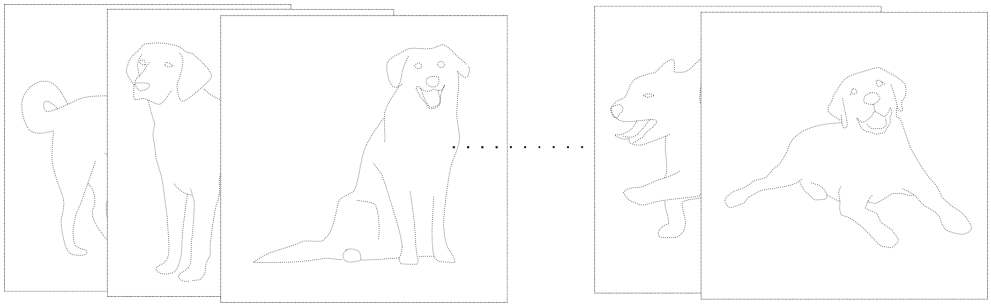
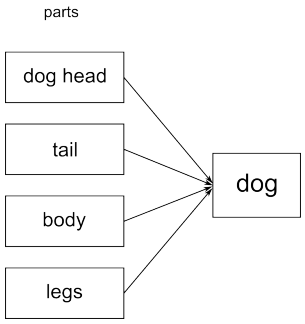


Figure 3.1.3: Multiple templates for several dog poses

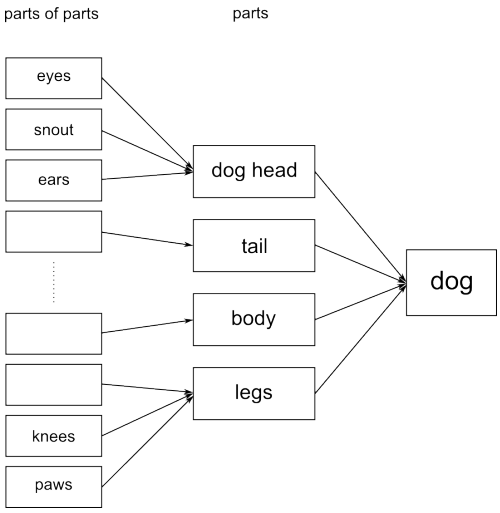
This is obviously a very unelegant solution, since we might require a large number of such templates, and even then we're not sure if we've covered all poses. What's also problematic is that we might require quite a lot of training data for each such template, which means we will have to collect a huge amount of images of dogs in any given pose.

Let's suggest a solution which is more invariant to multiple poses. Instead of detecting the whole dog, we might try to detect only the parts of the dog, such as the dogs head, the legs, the tail and the body, and loosely require that each part is attached to the body in some way. We can for instance then consider training smaller templates that only try to detect the parts, for instance a template that detects the dogs head, one for the tail, etc, which we pass over the images looking for a match. We can classify the image as containing a dog if enough loosely connected "dog-parts" are detected present in the image. We can sketch up our "dog parts" classifier as below:



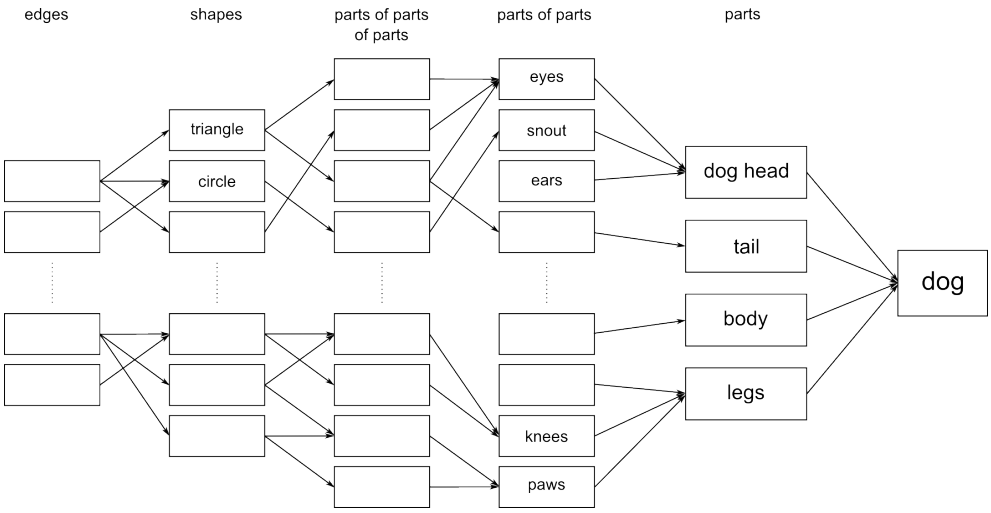
This will be more invariant to different poses, since we in a way ignore the "configuration" of the parts, and only care that they are present, and connected. This also makes it easier to train our model, since we don't need to have image examples of all the dogs poses, only examples of the dog parts.

Even so, we still have a problem with poses of the individual elements. A dogs head might be facing right, forward, or left, which makes it hard to use a simple template to match all dog heads. Taking our idea further we can break down this detector also into parts: individually detecting the parts of the dogs head such as ears, eyes, snout, tongue etc, and for the legs we could detect parts such as paws, knees etc. Using the same idea and training smaller templates for each part, we can sketch up our classifier as below with one more "layer":



This would hopefully be even more pose invariant, and we might need even fewer image examples, since we then no longer need examples of different dog head poses. A problem we might now consider, is that an eye or an ear might appear very

different depending on lighting, background, the quality of the image, fur color, etc. To create invariance to these kind of differences, we can build even more abstraction levels, detecting shapes such as circles for the eyes, triangles for the ears, all the way up to detecting simple edges, to ensure invariance to different poses, lighting, camera quality, etc. As we study our heuristic classifier, it is easy to imagine that many of the lower level templates can be shared, since shapes and edges are simpler than the higher level complex shapes, which means our classifier might end up looking something like this:



This kind of heuristic classifier illustrates that using layers of abstraction might make our classifier more robust, and also efficient to train, since we probably need fewer image examples. Furthermore, if we wanted to classify a relatively similar object, such as a horse, fox or bear, many of the higher level features, such as eyes, ears, legs, etc might be shared between the classifiers. This means that in a multi-class classification task, hierarchical layers of abstraction might be very efficient and allow sharing of information between tasks.

Incidentally, our heuristic classifier is very similar to what we will describe as deep convolutional networks - in fact, if we wanted a smooth detection indicator for each "part", a natural transformation is a logistic function, which is commonly used in neural networks (as we explain in section 3.2.1). Obviously, building such a model manually would require a lot of work. What deep learning tries to do, is to learn such a hierarchy of features directly from data, thus avoiding the tedious manual labor.

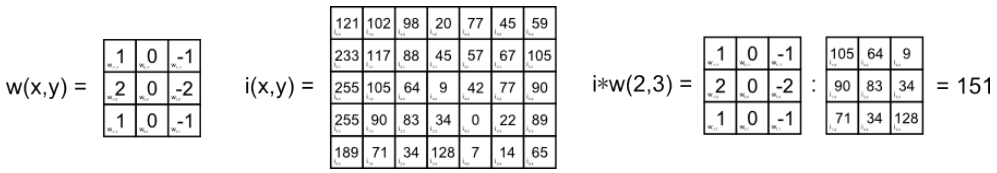


Figure 3.2.1: An example of a filter and an image and the convolution applied to a patch of the image. The symbol : indicates the frobenius inner product, i.e. summed elementwise multiplication.

3.2 Convolutional networks

In this thesis we will only describe the "archetypical" example of a deep learning architecture, a convolutional neural network (or CNN for short). Before delving into the details, we will give a brief description of the convolution operation used in convolutional neural networks (and in computer vision in general).

While the term convolution in mathematics refer to a more general operation, we will in this thesis only use the term to refer to a 2-dimensional discrete finite convolution:

$$(I * w)(x, y) = \sum_{m=-M}^{M} \sum_{n=-N}^{N} w(m, n) \cdot I(x + m, y + n)$$

where $I(x, y)$ is a mapping to some 2d matrix, commonly a representation of an image, and $w(x, y)$ is a mapping to a smaller $(2M + 1) \times (2N + 1)$ 2D matrix, called a "kernel" or a "linear filter". We will mostly use the term "linear filter" when describing this matrix. This convolution is equivalent to the frobenius product of the filter and a local patch in an image, see figure 3.2.1.

This operation can be applied to an entire image in order to for instance detect edges or presence of features, depending on the values of the filter. In figure 3.2.2 we show the effect of convolving a black and white image with a simple filter that detects vertical edges:

$$w(x, y) = \begin{bmatrix} -5 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 5 \end{bmatrix}$$

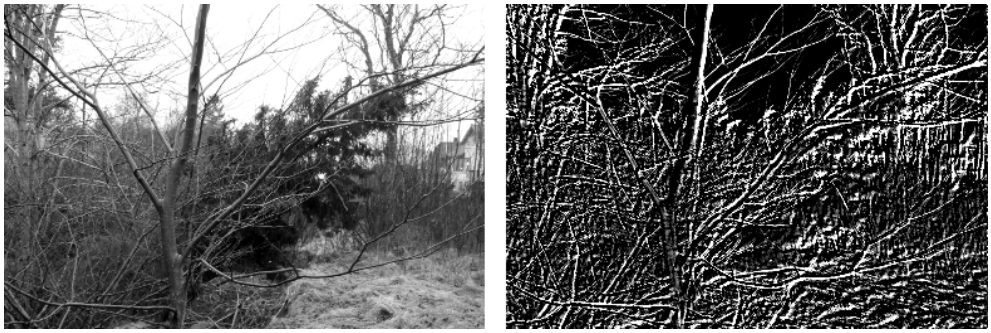


Figure 3.2.2: Left : unprocessed image, right : image convolved with the filter w(x,y)

Note that it is common to "pad" an image with zeroes, i.e. set $I(x, y) = 0$ for any points outside the image, but this varies with the task. Without such padding, the convolution is only well defined for points where the linear filter does not flow "outside" the image, and thus the output of a convolution over an $A \times B$ image will be a smaller image of dimensions $(A - 2M) \times (B - 2N)$.

Since we will be using inputs with more than one channel, such as color images, the convolution will actually be over three dimensions, but the idea remains largely the same, with the difference that the convolution will be summed over multiple channels:

$$(I * w)(x, y) = \sum_{c=1}^{C} \sum_{m=-M}^{M} \sum_{n=-N}^{N} w_c(m, n) \cdot I_c(x + m, y + n)$$

In convolutional neural networks, each layer consists of multiple linear filters which are convoluted over the input, and then transformed with a nonlinear function. Each filter uses the same weights over the entire input, so the effect is that each layer learns general linear filters which are translation invariant to the input. Since the high dimensionality of the data often poses challenges to training the networks, there are several strategies for dimensionality reduction commonly implemented in the network as well. One of these is applying the convolution in "strides", e.g. only applying the linear filter every fourth pixel for instance, which will have the effect of reducing the dimension of the input by four. Another commonly used strategy, is including a "sub-sampling" (or pooling) step, where the dimensionality of the output is reduced by passing on only the max (or average) of (usually non-overlapping) blocks of neighbouring inputs. This step also has the

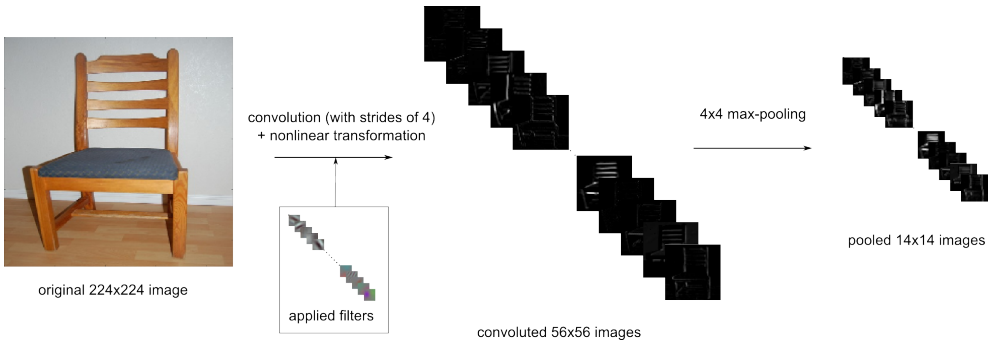


Figure 3.2.3: Example of first layer from our trained deep convolutional network

great advantage of inducing invariance to scale, translation and, to some degree, rotations. See figure 3.2.3 for an example of the output and filters of such a network.

It's easy to visualize the first layer, which tends to learn edge detection filters. However, due to the pooling and multidimensional nature of the higher layers, it's hard to directly view what features the next layers learn. Zeiler and Fergus (2014) recently created a method to show which pixels in an input image create the strongest activation in an intermediate layer. This allows us to find parts of images which create strong activations for a specific filter, and thus gives us an idea of what features the filters are trying to detect. In figure 3.2.4 below, they plot a selection of the activations for a number of layers in a trained model - the image patches to the right show parts of images with highest activations for that specific filter, while the grey patches to the left show the pixel activations for the corresponding image part.

From these visualizations we can see that each layer learns more and more complex and fine-detailed filters compared to the preceding layer. We can compare some of the learned features to the "parts" we described in our heuristic classifier above. The network analyzed by Zeiler and Fergus was trained to classify amongst other things several different breeds of dogs, and we can see in figure 3.2.5 that the network has actually managed to learn high-level features which detect parts such as "pointy ears", "legs", "dog faces", etc.

This shows that it is useful to think of deep convolutional networks as learning features which map to parts of an object. It should however be noted that since deep convolutional networks are discriminatively trained, they only learn the most

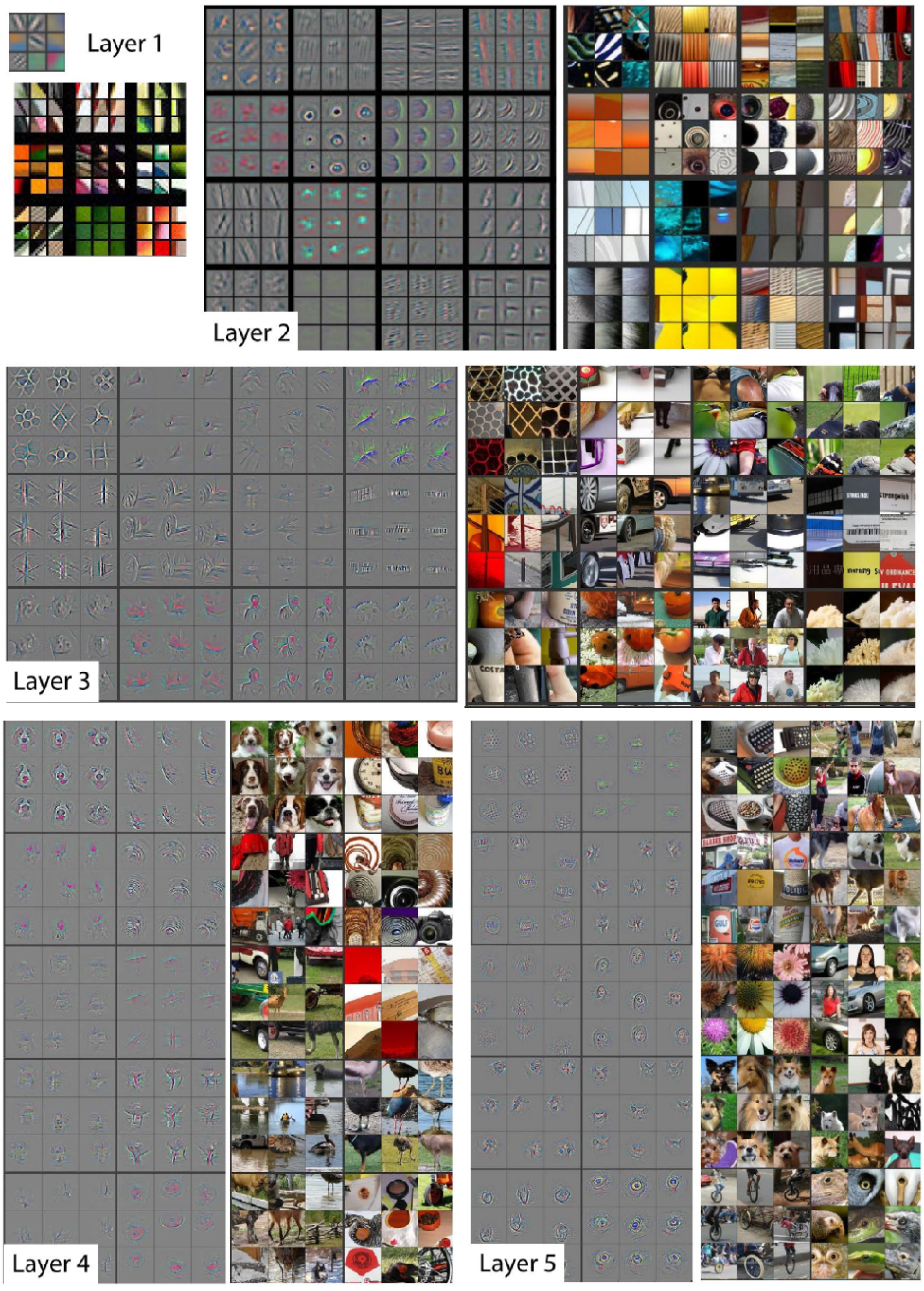


Figure 3.2.4: Filter examples from Zeiler and Fergus (2014)

Figure 3.2.5: Examples of learned "dog parts" in a deep convolutional network

discriminative parts of an object, and commonly also ignore the positioning of parts relative to each other.

Convolutional neural networks have actually been around for a long time, and similar models was introduced as early as 1989 by Yann LeCun, but due to problems training them (we'll get to why later), they haven't been used extensively until recently.

Though the earliest "deep learning" architectures were created for vision tasks, they've since been applied with great success to several other fields such as automated translation, speech recognition (Dahl et al., 2012), music analysis (Van den Oord et al., 2013) and natural language processing (Socher et al., 2011). The fields that gain most from deep learning architectures seem to be fields with highly structured data, i.e. fields where manual feature engineering has traditionally been very important.

Since we for our image analysis task use convolutional neural networks, we will use the next sections to go through the details of how the networks are put together and trained.

3.2.1 General Theory about (Convolutional) Neural Nets (and some history)

The convolutional neural net as described above, belongs to a larger family of models called *artificial neural networks*.

The family of neural networks grew out of research into artificial intelligence and theories about how the brain processes information, and the first such models, called perceptrons (Rosenblatt, 1958), were developed in 1950s as a reproduction of how nerve synapses in the brain communicate. Due to the development in a separate field, neural network theory often uses different terminology than commonly used in statistics, but we will mostly stick to the terminology used in classical statistics wherever possible.

Informally the family can be described as non-linear transformations that are chained together in a network. The non-linear transformations are usually defined as :

$$f(x) = \sigma(b + \sum_{i=1}^{N} w_n \cdot x_n)$$

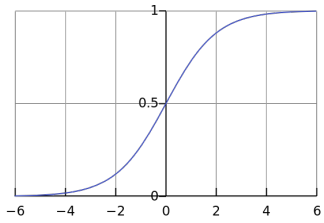


Figure 3.2.6: The logistic function, $\sigma(x) = \frac{1}{1+\exp(-x)}$,commonly used as activation function in neural networks

where $\sigma(\cdot)$ is called the activation function, usually chosen to be the logistic function (see figure 3.2.6), b and $(w_1, w_2, , w_N)$ are parameters of the model to be learned during training, commonly called the bias and weights respectively, and x is the vector of inputs.

A very basic example of such a model with one input and a probabilistic output (for binary classification) is this:

$$y(x) = \sigma(b_3 + \sum_{i=1}^{2}[w_{i+2} \cdot \sigma(w_i \cdot x + b_i)]) \tag{3.1}$$

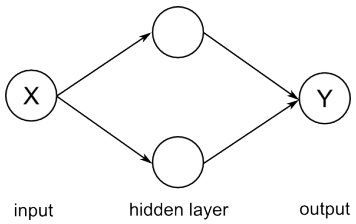


Figure 3.2.7: Figure of the neural network in formula 3.1

Neural networks are often illustrated with graphs such as the one in figure 3.2.7, where the nodes (also called *units*) represent the transformed version of their inputs. Each series of transformations is called a *layer*, and thus such a model could be called a 2-layer neural network, with the output as the second layer. The transformed values in the middle layer can be viewed as unobserved latent factors of the model, which is why these layers are often called "hidden" layers.

To simplify explanations, we introduce a simple multi-layered network, see figure 3.2.8, which we will use as an example in descriptions going forward. We also

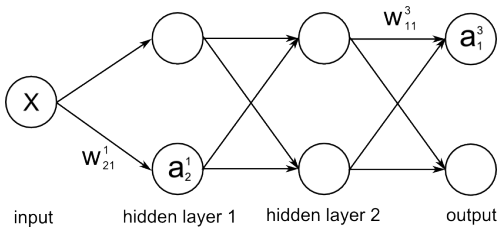


Figure 3.2.8: Example multi-layered neural network

introduce some general notation that will come in handy later.

We will describe each layer by a number l, so that the "input layer" is layer 0 and the "output layer" is layer 3. For each node n in layer l we will describe the output (or the activation) from the node as a^l_n, and the pre-activation-function output as z^l_n. We furthermore describe the *bias* term in node n in layer l as b^l_n and the weight for the output from node m in the preceding layer as w^l_{mn}. We thus have for each node in layers with $l > 0$:

$$z^l_n = b^l_n + \sum_{m=1}^{2} w^l_{mn} \cdot a^{l-1}_m$$

$$a^l_n = \sigma(z^l_n)$$

To enable us to formulate the model using linear algebra, we define furthermore a^l as the vector of activations at layer l, and z^l as the vector of pre-activation outputs. We also denote the matrix W^l as all the weights for layer l, with row m and column n containing the weight w^l_{mn}. We can then easily formulate a single layer of this network as:

$$a^l = \sigma(W^l \cdot a^{l-1} + b^l)$$

For the hidden layers we use as activation function $\sigma(\cdot)$ the regular logistic function. Note here that we use $\sigma(\cdot)$ to denote the element-wise logistic function applied to the input vector. For the final layer we use the softmax function as activation function:

$$\sigma(z^L) = \left(\frac{\exp(z^L_1)}{\sum_{i=1}^{N} \exp(z^L_i)}, \ldots, \frac{\exp(z^L_N)}{\sum_{i=1}^{N} \exp(z^L_i)} \right)$$

which is similar to the output from a multinomial logistic regression, and allows us

to get class probabilities for (in our case) two classes. Note that the model predictions for class n in our case is the activations in the final layer, a_n^3, while the input is a_1^0.

Furthermore, if we define $f_l(X) = \sigma(W^l X + b^l)$ we can then formulate our simple model as:

$$a^3 = f_3(f_2(f_1(x))) = f_3 \circ f_2 \circ f_1 \circ x$$

In this way we see that multi-layer neural networks generally are chained nonlinear compositions of the input. We can also mention that since z^l are affine transformations of the input, these kind of networks can be viewed as recursive generalized linear models with $\sigma(\cdot)$ as link functions.

In the case of the convolutional networks mentioned earlier, the dimensionality of the inputs and weights are larger. Since each layer output is represented by a $m \times m$ matrix, we have that both a^l and z^l are $n^l \times m \times m$ matrices, where n^l is the number of "filters" in layer l. Furthermore, the weight matrix W^l is a $n^l \times n^{l-1} \times o^l \times o^l$ matrix (or rather, tensor), where o^l is the size of the "filters" in layer l.

Altogether we have that:

$$z_{nij}^l = b_n^l + \sum_{a=0}^{O-1} \sum_{b=0}^{O-1} \sum_{c=1}^{N^{l-1}} w_{ncab}^l \cdot a_{c(i+a)(j+b)}^{l-1} \tag{3.2}$$

and as before we have that $a_{nij}^l = \sigma(z_{nij}^l)$. See figure 3.2.9 for an illustration of the convolution operation.

3.2.2 Training Neural Networks

Neural networks are generally fitted by some kind of gradient descent over a loss function $L(\theta)$ chosen for the task. In our case, we use the log-likelihood loss, which is a natural choice similar to the optimization objective of a multinomial logistic regression:

$$L(\theta) = -\frac{1}{N} \sum_{n=1}^{N} \log(p(c = k_n | I_n, \theta))$$

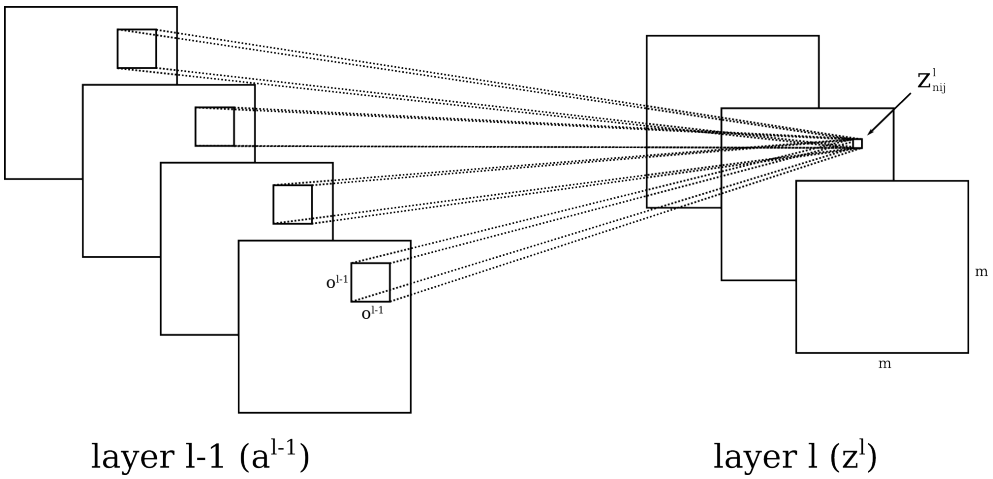


Figure 3.2.9: An illustration of the convolution operation in (3.2). Note that the weight matrix W^l is not illustrated.

where k_n is the true class of image n, and $p(c = k_n|I_n, \theta)$ is the estimated probability of image n being from class k_n according to our model with parameters θ, where θ is the set of all weights and biases of the convolutional network. Gradient descent iteratively updates the network parameters θ by following the gradient of $L(\cdot)$ at the current parameter estimate either until the updates are smaller than some threshold, or until the loss no longer shrinks on some held-out test set:

$$\theta_{t+1} = \theta_t - \alpha \cdot \nabla L(\theta_t)$$

where $\alpha > 0$ is the learning rate. Since calculating the true gradient can be very costly when training on a large dataset, deep neural networks are almost always trained with stochastic (batch) gradient descent. This is a variation of gradient descent where we instead of calculating the true gradient over the entire dataset on each iteration, calculate the gradient over only a small batch of training data. Commonly, the training data is first randomly shuffled, and the parameters are iteratively updated by passing over the dataset, possibly several times. It's also common to include momentum in the updates in order to smooth the parameter updates :

$$\theta_{t+1} = \theta_t + V_{t+1}$$

$$V_{t+1} = \mu V_t - \alpha \cdot \nabla L(\theta_t)$$

Here μ in $[0,1]$ is a parameter which controls the strength of the momentum, i.e. how much of the previous update we include in the current update. This ensures that the updates are not too sensitive to the individual batch gradients (which may be noisy) and makes the convergence more stable.

The starting value of the network parameters θ_0 has proven to be very important to ensure convergence to a good minimum in deep neural networks (see e.g. Sutskever et al. (2013)). We'll discuss this in further detail later, but we briefly mention here that the weights of the network typically are randomly initialized with a Gaussian with mean zero and variance either fixed or dependent on the network architecture. To ensure convergence, the learning rate α also needs to decrease as we iterate over the training set. There are several strategies for how to do this, but a common method in training convolutional networks is to initially set the learning rate to a low value, e.g. 0.01, and decrease it by a constant factor during training whenever the loss (evaluated on some held-out test set) seems to plateau.

So far we haven't discussed how the gradient in the formula above is calculated for neural networks. In the next section we will describe an algorithm which allows us to efficiently evaluate the gradients.

3.2.3 Backpropagation

Our intention is to calculate the gradients $\nabla L(\theta)$, where $\theta = (w^1_{11}, w^2_{11}, w^1_{12}, w^2_{12}, b^l, \ldots, b^L)$ and thus $\nabla L(\theta) = (\frac{\partial L}{\partial x^1_{11}}, \ldots, \frac{\partial L}{\partial b^L})$. We will derive the backpropagation algorithm for our generic example above in order to illustrate the principle, and then show that this carries over without large modifications to the convolutional network we use.

First we note that the loss we use, the log-likelihood, is a sum of the log-likelihood for each individual sample, which means that we can calculate the gradients for each sample independently, and then average the gradients over all of our examples to get the final gradient.

We first note that by the chain rule, we have that:

$$\frac{\partial L}{\partial w_{ij}^l} = \frac{\partial L}{\partial z_i^l}\frac{\partial z_i^l}{\partial w_{ij}^l} = \frac{\partial L}{\partial z_i^l}a_j^{l-1} \tag{3.3}$$

$$\frac{\partial L}{\partial b_i^l} = \frac{\partial L}{\partial z_i^l}\frac{\partial z_i^l}{\partial b_i^l} = \frac{\partial L}{\partial z_i^l} \tag{3.4}$$

thus in order to calculate the gradients we (only) need to calculate each of the "activations" a_j^l, and the terms $\frac{\partial L}{\partial z_i^l}$. We first calculate this term for the last (output) layer:

$$L = -\log\left(\frac{\exp(z_y^3)}{\sum_{j=1}^2 \exp(s_j^3)}\right)$$

$$\begin{aligned}\frac{\partial L}{\partial z_i^3} &= -\frac{\sum_{j=1}^2 \exp(z_j^3)}{\exp(z_y^3)}\left[1_{i=y}\cdot\frac{\exp(z_y^3)}{\sum_{j=1}^2\exp(z_j^3)} - \frac{\exp(z_y^3)\cdot\exp(z_i^3)}{(\sum_{j=1}^2\exp(z_j^3))^2}\right] \\ &= \frac{\exp(z_i^3)}{\sum_{j=1}^2\exp(z_j^3)} - 1_{i=y} = a_i^3 - 1_{i=y}\end{aligned} \tag{3.5}$$

We can thus see that $\frac{\partial L}{\partial z_i^3}$ can be seen as the *error* between our prediction and the truth. For this reason $\frac{\partial L}{\partial z_i^3}$ is often called either the "error" or the "delta" and denoted by δ_i^l. We now derive a recursive version of this term for the other layers via the chain rule:

$$\begin{aligned}\frac{\partial L}{\partial z_i^{l-1}} &= \frac{\partial L}{\partial a_i^{l-1}}\frac{\partial a_i^{l-1}}{\partial z_i^{l-1}} = \frac{\partial L}{\partial a_i^{l-1}}\sigma'(z_i^{l-1}) \\ &= \left(\sum_{j=1}^N \frac{\partial L}{\partial z_j^l}\frac{\partial z_j^l}{\partial a_i^{l-1}}\right)\sigma'(z_i^{l-1}) = \left(\sum_{j=1}^N \frac{\partial L}{\partial z_j^l}w_{ij}^l\right)\sigma'(z_i^{l-1})\end{aligned} \tag{3.6}$$

Thus we see that we can calculate the error terms $\frac{\partial L}{\partial z_i^{l-1}}$ in the preceding layers as a function of the error terms $\frac{\partial L}{\partial z_i^l}$ in the current layer. This allows us to first do a "forward pass" to calculate all the activations a_i^l and z_i^l, then calculate the errors at the last layer and "propagate" the errors layer by layer back through the net (which is why it's called backpropagation) to calculate the errors for the preceding layers. Finally, we can calculate the gradients from our calculated errors and activations

using the formulas in 3.3 and 3.4 above. This allows us to very efficiently evaluate the gradient on each example, then average over all the examples to get the final gradient. Note that this generalizes to larger networks with other loss functions, as long as the losses are sums of the individual losses.

This also generalizes relatively simply to our convolutional network, but to prove that this is so we show how to calculate the error terms for our convolutional network. Note that in this case, the gradient wrt the weights is a sum of the gradients over the entire input (image), since the weights are convoluted over the input:

$$\frac{\partial L}{\partial w^l_{ncab}} = \sum_{i=1}^{M^l}\sum_{j=1}^{M^l} \frac{\partial L}{\partial z^l_{nij}} \frac{\partial z^l_{nij}}{\partial w^l_{ncab}} = \sum_{i=1}^{M^l}\sum_{j=1}^{M^l} \frac{\partial L}{\partial z^l_{nij}} a^{l-1}_{c(i+a)(j+b)}$$

but as before we only need to calculate the activations and the error terms to get the weight gradients. We calculate the error terms:

$$\frac{\partial L}{\partial z^l_{nij}} = \frac{\partial L}{\partial a^l_{nij}} \frac{\partial a^l_{nij}}{\partial z^l_{nij}} = \frac{\partial L}{\partial a^l_{nij}} \sigma'(z^l_{nij})$$

$$\frac{\partial L}{\partial a^{l-1}_{nij}} = \sum_{a=0}^{O^{l-1}}\sum_{b=0}^{O^{l-1}}\sum_{c=1}^{N^{l-1}} \frac{\partial L}{\partial z^l_{c(i-a)(j-b)}} \frac{\partial z^l_{c(i-a)(j-b)}}{\partial a^{l-1}_{nij}} = \sum_{a=0}^{O^{l-1}}\sum_{b=0}^{O^{l-1}}\sum_{c=1}^{N^{l-1}} \frac{\partial L}{\partial z^l_{c(i-a)(j-b)}} w^l_{cnij}$$

$$\frac{\partial L}{\partial z^l_{nij}} = \sum_{a=0}^{O^{l-1}}\sum_{b=0}^{O^{l-1}}\sum_{c=1}^{N^{l-1}} \frac{\partial L}{\partial z^l_{c(i-a)(j-b)}} w^l_{cnij} a^{l-1}_{c(i+a)(j+b)}$$

thus we can back-propagate errors as before (though with some more work).

Our convolutional neural network also includes max-pooling layers. These are non-parametric deterministic functions of the preceding layers, where we only pass on the the largest activations in a block in the preceding layer, so in the backpropagation algorithm, we do a forward pass as before, but we only propagate the error back through the activations "selected" by the max-pooling. In this way the max-pooling makes backpropagation of errors sparser in earlier layers.

The discovery of the backpropagation algorithm in 1986 (Rumelhart et al., 1988), spawned a flurry of interest in neural networks during the late eighties. Inspired by how the neural cortex processes information, LeCun et al. (1989) developed the first convolutional network, *LeNet*, which was used to classify

handwritten numbers. The trained model managed to achieve state-of-the-art results on the task, and is still in use for interpreting written text. However, the model was finicky and took a long time to train, even on a dataset of small size by today's standards, which hindered its use in other applications.

The reason why training deep neural networks was so difficult was unclear for a while, until Sepp Hochreiter explained in his 1991 thesis (Hochreiter, 1991) that this was due to the problem of *vanishing gradients*. In the next section we will describe this problem in a bit more detail.

3.2.4 Unstable/vanishing gradients in deep neural networks

The *vanishing gradient* problem describes the fact that with regular deep neural networks, the gradients will tend to shrink as we propagate the errors back through the layers, which means an early layer will have an exponentially smaller gradient than the layer after it. To show the reason for this, we inspect the gradients for our example network from earlier:

$$\begin{aligned}
\frac{\partial L}{\partial b_i^l} &= \frac{\partial L}{\partial z_i^l} = \left(\sum_{k=1}^{N} \frac{\partial L}{\partial z_k^{l+1}} w_{ik}^{l+1} \right) \sigma'(z_i^l) = \left(\frac{\partial L}{\partial z^{l+1}} \right)^T W_{i\cdot}^{l+1} \sigma'(z_i^l) \\
\frac{\partial L}{\partial b_i^{l-1}} &= \frac{\partial L}{\partial z_i^{l-1}} = \left(\sum_{k=1}^{N} \frac{\partial L}{\partial z_k^{l}} w_{ik}^{l} \right) \sigma'(z_i^{l-1}) = \left(\frac{\partial L}{\partial z^{l}} \right)^T W_{i\cdot}^{l+1} \sigma'(z_i^{l-1}) \\
&= \left(\frac{\partial L}{\partial b_i^{l}} \right)^T W_{i\cdot}^{l+1} \sigma'(z_i^{l-1})
\end{aligned}$$

As we see, the gradient of the bias in layer $l-1$ is a factor of the gradients of the biases in the preceding layer l. However, in the case we use the logistic function as activation function, we can see in figure 3.2.10 that the term $\sigma'(x)$ is always smaller than $\frac{1}{4}$.

This means that unless the weights in the term $W_{i\cdot}^{l}$ are large enough to counter the diminishing effect of $\sigma'(x)$, our gradient will be a factor smaller than the gradient in the layer preceding it, and in further preceding layers, the gradients will be many factors smaller. As it happens, the weights usually tend not to be large enough to counter this effect, so we get *vanishing gradients* in the early layers. In the example above we only showed this for the gradient for the biases b^l, but we get a

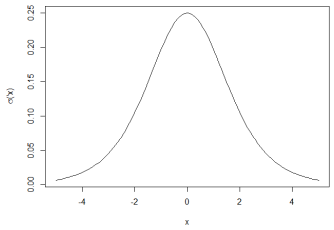


Figure 3.2.10: Derivative of logistic function

mostly similar situation with the gradient for the weights W^1. One could imagine a scheme where we try to make the weights large enough to counter this effect, but this is a bad idea, since we then risk to run into the opposite effect, exploding gradients, where the gradients in the early layers grow exponentially larger as they are propagated back through the network, which makes gradient descent extremely unstable and unlikely to converge. Another reason this is a bad idea is that a common regularization scheme (which we will discuss later), is L2-regularization of the weights, i.e. *forcing* the weights to be small.

The vanishing gradient effect is a considerable problem, since this means that training the first layer will take longer and longer time the more layers we add. In other words, deep neural networks using the logistic function as activation function generally have problems with unstable gradients, and avoiding these problems requires extremely careful monitoring and initialization of the weights, which have made the traditional deep neural networks so finicky and time-consuming to train.

This problem caused interest in deep neural networks to slow down considerably during the nineties and early oos, since methods that were significantly easier to train, such as support vector machines, usually reached equivalent precision. During mid-2000 a lot of research was put into clever initialization of these deep networks via unsupervised pre-training. However, recent modifications have significantly improved the ease of training deep learning neural nets without needing such pre-initialization, and we'll describe these in the next three sections.

3.2.5 Rectified Linear Units

Instead of the logistic function, which was the source of the vanishing gradient issue, researchers started around 2012 experimenting with using the Rectified

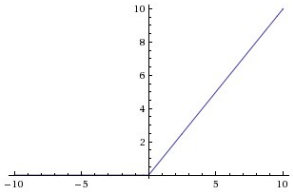


Figure 3.2.11: Rectified linear function

Linear function as activation function instead:

$$\sigma(x) = \max(x, 0)$$

It might seem odd to use this function, since it's non-differentiable at zero, i.e.:

$$\sigma'(x) = \begin{cases} 1 & x > 0 \\ 0 & x < 0 \end{cases}$$

While this is a significant analytical problem, it doesn't matter in practice, since the function is differentiable at any point arbitrarily close to zero and the function is unlikely to be active at exactly zero.

A significant *advantage* of this function within the context of deep neural networks, is that the derivative is constant when the function is "active". This means that we avoid the issue with vanishing gradients we saw earlier. A second advantage is that this function induces sparse forward flow in the neural network : if nodes are not active, they are simple set to zero, which means the forward calculations often are faster. The overall result is that deep neural networks with rectified linear functions are faster to evaluate and many times faster to train.

3.2.6 (Dropout) regularization

Neural networks are very flexible models, and thus they are easy to overfit. A common regularization method traditionally used for neural networks (and for many other machine learning methods) is L2-regularization of the weights, commonly called weight decay in the context of neural networks. This merely means we add an extra term to our cost function that forces weights to not grow too large:

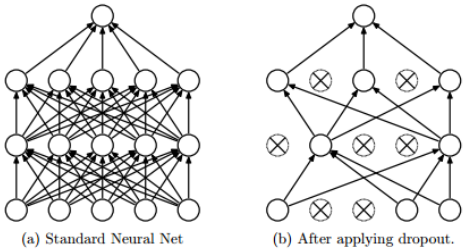


Figure 3.2.12: Dropout illustration (from Srivastava et al. (2014))

$$\hat{L(\theta)} = L(\theta) + \lambda \cdot \sum_{w} (w_{ij}^{l})^2$$

where λ is a parameter which controls the strength of the regularization.

However, a very successful regularization method that was invented recently, is the so-called *dropout* regularization method, which was introduced by Hinton et al. (2012).

The method consists of randomly dropping out units in a layer during training. For a batch of examples, we randomly drop out any unit in the hidden layers with probability p, and then do backpropagation and gradient descent as usual. This means that for each batch, we train a random subset of the weights, each with a random subset of inputs. At prediction time, we include all trained units as usual. Note that each node was trained with only approximately p of the inputs present at any time, so when we include all inputs, the activations may be much larger than what they were during training. To counter this, we multiply the weights by p at prediction time.

This type of regularization can loosely be seen as a form of *bootstrap aggregation*. Bootstrap aggregation is a regularization technique where we create many sets of training data by bootstrap sampling from the original training set, and then train a model on each of these sets. At prediction time, we average the predictions across all these models. This has been shown to be a very effective way of reducing variance of some models, and thus often reducing the prediction errors made by the models. Such a scheme, though possible for deep neural networks, would be very time-consuming, since deep neural networks usually take a long time to train. Instead, dropout is a form of approximate bootstrap aggregation, since we are in effect training on a huge set of randomly perturbed models, and ensembling the

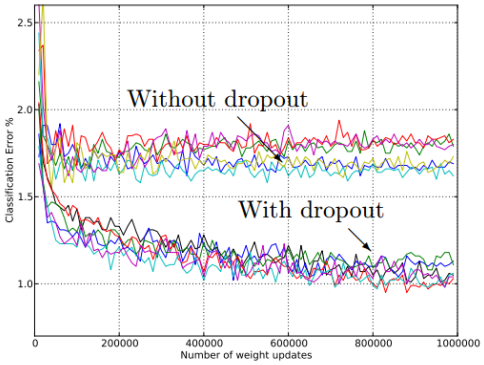


Figure 3.2.13: Effect of dropout on training error (from Srivastava et al. (2014))

models (or in our case, weights) at prediction time. Similar to bootstrap aggregation, we can see dropout regularization as a way of creating "noise" in the training data, so that the trained layers (and the model overall) are more robust to small variations in the input.

Empirically, dropout has been shown to perform surprisingly well, and often much better than networks regularized with L2-regularization alone. In practice, dropout tends to be used in conjunction with L2-regularization, since both methods seem to independently reduce the risk of overfitting. Note that the probability of dropping out any unit, p, is usually set to 0.5, which seems to perform very well for most networks, but this is a actually hyperparameter of the model which can be tuned based on a held-out test set.

Recently Gal and Ghahramani (2015) showed that dropout as applied in neural networks, is equivalent to a variational approximation to a bayesian neural network with Gaussian processes as priors. They further suggest that this explains why dropout has a regularizing effect, as bayesian models are less susceptible to overfitting. Their interpretation also suggest several interesting routes for probabilistic inference, e.g. making uncertainty estimates for classification results, but we unfortunately did not get time to explore this in our thesis.

3.2.7 Initialization

A good initialization of the parameters are usually important for proper convergence in non-convex optimization tasks. This holds for stochastic gradient

descent for deep neural networks as well. For the initial guess of the weights, we'd like the resulting activations to generally be of the same scale as the "signal" flows through the network, to ensure proper convergence. If the initial weights are too small, the activations will be smaller and smaller for each layer, and learning will tend to stall, while if the initial weights are too big, the activations will grow for each layer, and learning may diverge. With activations that are of different scales, setting the learning rate also becomes hard, as it is very hard to set a learning rate that works uniformly well for all weights.

A common setting for initializing deep neural networks has been to draw the weights from a gaussian with mean 0 and fixed variance, for instance 0.01. This works reasonably well, but does not solve the problem mentioned above, and we usually run into problems with convergence as we add more and more layers to the network. To alleviate this problem, Glorot and Bengio (2010) suggested an alternative initialization of the weights where the variance is not fixed, but depend on the number of inputs to each unit. We will discuss the main points of the derivation here.

Glorot suggested assuming that activations are linear, i.e. $Y = b + WX$. This assumption holds approximately when we use the logistic function as activation function, since the logistic function is close to linear when the input is close to zero. Furthermore, they assume that the input signal X has mean 0, and the W_i are independent from each other as well as independent from the X_i. We then have that:

$$\begin{aligned}
\operatorname{Var}(Y) &= \operatorname{Var}\left(b + \sum_{i=1}^{n} X_i W_i\right) = \sum_{i=1}^{n} \operatorname{Var}(X_i W_i) \\
&= \sum_{i=1}^{n} \left[\mathrm{E}[X_i]^2 \operatorname{Var}(W_i) + \mathrm{E}[W_i]^2 \operatorname{Var}(X_i) + \operatorname{Var}(X_i) \operatorname{Var}(W_i)\right] \\
&= \sum_{i=1}^{n} \operatorname{Var}(X_i) \operatorname{Var}(W_i) = n \operatorname{Var}(X) \operatorname{Var}(W)
\end{aligned}$$

To have activations of similar scale through the network, we would like the variance of the input X and output Y to be roughly the same, i.e. we would like $n \operatorname{Var}(W) = 1$. According to these assumptions, a reasonable choice for the variance of the weight initializations is $\operatorname{Var}(W) = \frac{1}{n_i}$, where n_i is the number of

inputs to the unit. This initialization has empirically been shown to give much better convergence than initializations with fixed variance, and has since colloquially become known as *Xavier* initialization (from the main author Xavier Glorot).

This derivation was originally made based on the logistic activation function, and the assumptions of linear activations do not hold as well for the rectified linear activation. Nevertheless, Xavier initialization has empirically been shown to work well for rectified linear activations as well. Since then, He et al. (2015) has shown that proper initialization for ReL actually is very similar to Xavier, but we intuitively have to double the variance to counter the fact that ReL is only linear in half of output, i.e. $\text{Var}(W) = \frac{2}{n_i}$. We won't cover the derivation here, but refer to their paper for a proper discussion.

Altogether, rectified linear units, dropout regularization, and proper initialization of the weights, has made training deep neural networks much faster and more robust than earlier. In addition, increases in processing power, as well as increases in the amount of data, has made training deep neural networks both feasible and worthwhile.

3.2.8 Side note : The Imagenet Large Scale Visual Recognition Challenge

Since we will refer to it as we proceed, we briefly describe the image classification competition called the *Imagenet Large Scale Visual Recognition Challenge* (or *ILSVRC* for short).

The ImageNet database is a research database of hand-labeled images corresponding to distinct concepts and categories, such as "cars", "flowers" or highly specific classes such as "Scotch terrier", hosted by Stanford University. Since around 2010, a competition called the ILSVRC, based on the ImageNet database, has been hosted by Stanford university to foster developments in image recognition. The challenge is to correctly classify as many as possible images from a testset provided, with a classifier trained on a publicly available training set of approximately 1.5 million images from 1000 distinct classes, spanning from animals to man-made objects. The images are sourced from the world wide web and are of very varying quality, and thus the task can be said to be very challenging.

Since this challenge has been held every year since 2012 with the same training set (but with a new test set every year), the imagenet challenge has become a benchmark for comparing image classifiers, and performance on ILSVRC is widely cited in academic papers. It has also become common to share the architectures of the winning solutions to the public after each competition, and so a large number of image classifiers based on these architectures and trained on the imagenet dataset has been made publicly available. For our deep convolutional network architecture we will use two architectures which performed well in ILSVRC in 2012 and 2014 respectively, *AlexNet* and *GoogLeNet*.

3.2.9 Architectures

We will describe two deep neural network architectures which we tested for the image classification task. Both are architectures tailored for image classification, but differ in structure, depth, parameter size and accuracy. We will go through some of the motivation for the differences.

Both architectures consist of the building blocks we've described earlier, i.e. convolution layers, pooling layers, and regular "fully connected" layers, but they occasionally also apply something called "local response normalization". This is a form of normalization of the inputs which serves to keep activation in different filters on somewhat same level. Normalization was more important when using logistic activations, but it seems to help a little bit even when using rectified linear activations, probably for convergence reasons, which is why they're still included in these architectures. We refer to chapter 3.3 in Krizhevsky et al. (2012) for details.

The general structure of these deep networks, is that they apply several convolutions in series, which function as feature extractors. At the end of the network, there are some final layers which use the extracted features to classify the images. We illustrate the structure of both networks in figure 3.2.16 below.

The first architecture we test out is called *AlexNet*, referring to the author Alex Krizhevsky. This architecture mainly follows the structure suggested in Yann LeCun's *LeNet-5* model, by a series of initial alternating convolution and max-pooling layers, followed by a series of several convolution layers and finally two regular fully-connected layers and a final softmax layer which produces the classification output. The main difference between this architecture and LeNet-5, is

that it has a few more convolution layers, uses rectified linear activations for all convolution layers instead of logistic activations, and uses dropout to regularize the last fully connected layers. The approximate amount of parameters in this model is a whopping 62 million, which underlines the importance of proper regularization.

This model won ILSVRC 2012, reducing the state-of-the-art error rate from 26.2% to 15.3%, at the time a sensational improvement which served to reignite interest in the field of deep learning. This model has since served as a baseline reference for deep learning architectures in image classification tasks.

The second architecture we test out is called GoogLeNet, a reference to the team who made it, Google, and the first convolutional net, LeNet. This architecture tries to increase the depth of the network, thereby increasing discriminative capacity, while alleviating some perceived problems with convolutional architectures in general.

A perceived problem with convolutional networks, is that the learned filters are linear. Lin et al. (2013) suggest that this may decrease the capacity to flexibly represent features, and instead suggest exchanging the linear filters with a non-linear approximator such as a multilayer neural network, effectively creating a "network in network" structure. A simple way to do this is to prepend a regular convolution layer by another 1×1 convolution layer, which simply serves as a non-linear "preprocessing" of the input. GoogLeNet integrates this kind of structure at several points in their architecture, but also set this 1×1 convolution layer to have fewer outputs than inputs, thereby forcing the layer to learn a dimensionality reduction of the data. It might seem odd to call this a convolution layer, since it actually is not a proper convolution and only serves as a learned nonlinear factorization of the input filters, but this allows us to see the network as a structure of simple parts.

Another problem they focus on, is that the fixed size of filters may cause problems with learning features at differing scales. To accommodate this, google uses 1×1, 3×3 and 5×5 convolutional layers in parallel and simply concatenate the output from each of them. They also include a max-pooling layer in parallel, since max-pooling has been shown to reduce translational variance. Using this many convolutional layers in parallel could easily lead to a blow-up in the computational complexity, so they use the "network-in-network" 1×1 convolution layers as preprocessing "compression" steps for the 3×3, 5×5 layers, and

post-processing "compression" steps for the max-pooling layers.

They put all of this together in modules they call "Inception" modules (as a reference to the movie "Inception", where the main protagonist has a dream inside a dream), which they then stack as regular convolutional layers, see figure 3.2.14.

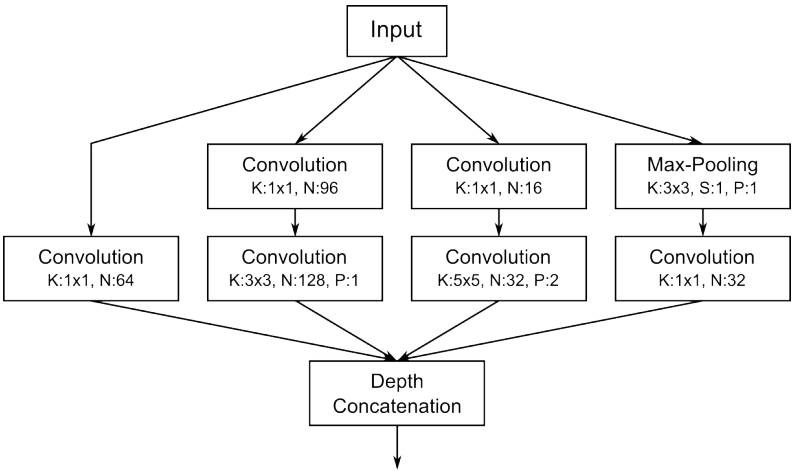


Figure 3.2.14: Inception module architecture

All the inception modules in the GoogLeNet architecture have the same structure, but only differ in the number of output from each parallel convolution layer. Therefore, in the overview of the GoogLeNet architecture, we only represent these modules by a block with numbers representing the outputs of each part, see figure 3.2.15.

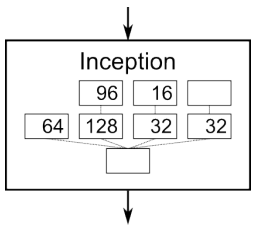


Figure 3.2.15: Inception module representation
The numbers represent number of outputs in the convolutional layers

Another problem Lin et al pointed out, is that in AlexNet, the final two fully connected layers stand for a full 88% of all parameters in the model due to the dense connections. This causes problems of overfitting and hampers effective learning. The original motivation of these layers are to be able to combine the

information from the learned representations in a proper way for classification, since inputting the learned representations directly to a softmax layer does not give good results. They then conjecture that using more flexible filters, such as the network-in-network structure, as well as adding more layers, will allow the learned representations to themselves serve as direct representations of the confidence in the categories of objects we try to classify, and that we thus need only an averaging over the final representations as input to the softmax layer. This also gives better interpretability, as it is then straightforward to see the final representation layers as confidence maps for specific classes of objects. This idea has been applied to the GoogLeNet architecture.

The final version of the network is 22 layers deep, which is a magnitude deeper than most networks, and can cause problems with learning in the early layers. Since the lower layers in the network should be reasonably good representations in themselves, they add classifiers at ⅓ and ⅔ of the network, which are only used during training, and serve to properly propagate loss to the early layers. During training, the loss from these classifiers are used for gradient updates as usual, but are weighted down relative to the final softmax classifier.

All in all, by increasing the flexibility of the convolutional layers at the same time as avoiding dimensionality blow-up, they manage to create a very deep network which is able to efficiently learn without overfitting. The final network has 6.8 million parameters, which is 12 times less than AlexNet, but has a modest increase in computational complexity due to the increased number of convolutional elements. The model won the ILSVR 2014 challenge, reducing the state-of-the-art error on the same task as AlexNet to 6.67%, showing that the added flexibility and depth of this model was worthwhile.

Note that in figure 3.2.16, all activations are rectified linear activations except the final layers, which are softmax. Unless noted, the stride is 1 and padding is 0.

3.2.10 Why does deep learning work so well?

Despite generating state-of-the-art performance on a large number of challenging tasks, the reason why deep learning performs so well is not thoroughly understood. Much of the work around neural networks and deep convolutional neural networks were motivated by research into how neural structures such as the visual cortex

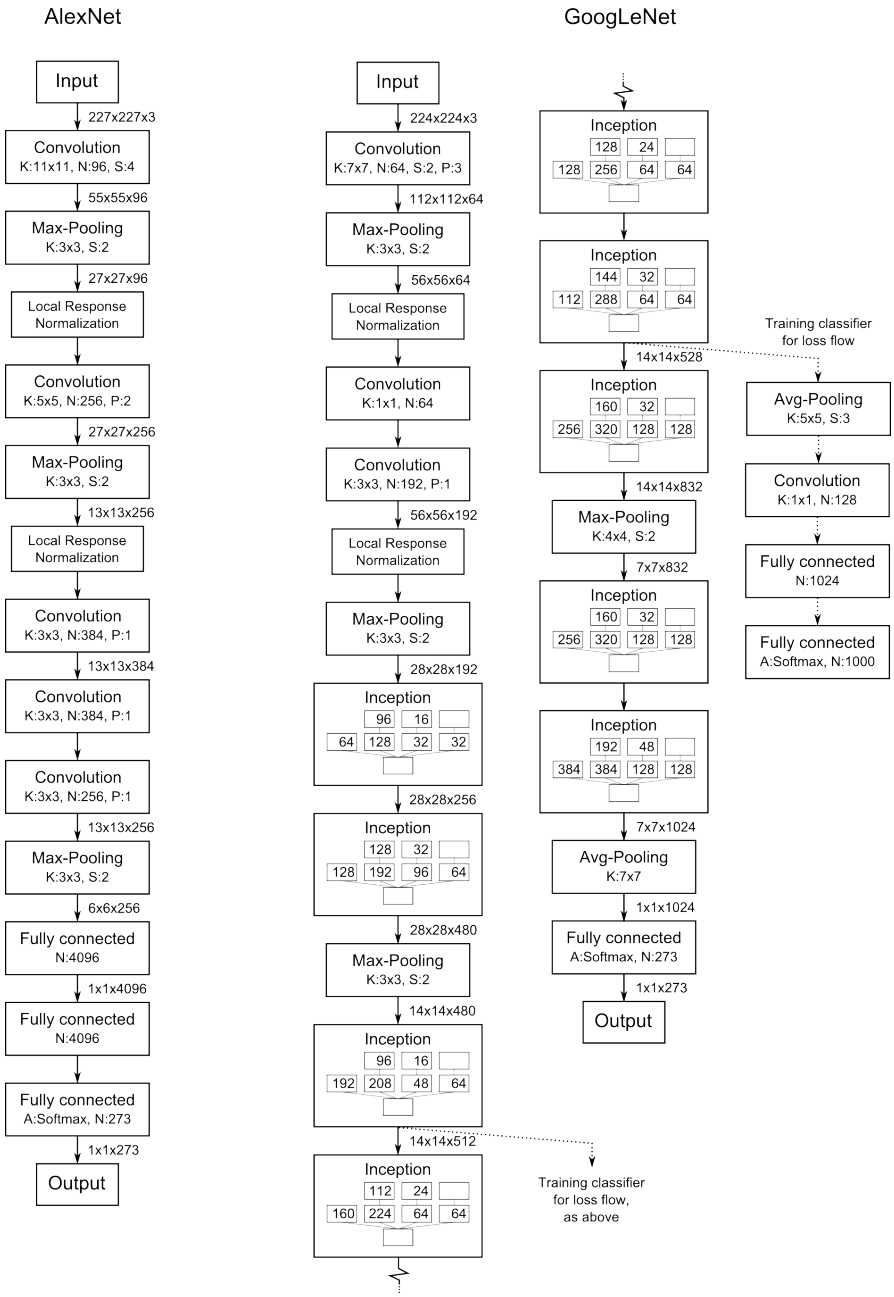


Figure 3.2.16: Architectures of AlexNet and GoogLeNet
N is number of outputs, K is kernel size, S is stride size, P is padding, the numbers between layers represent the dimension of the output

function (see e.g. Fukushima (1975)), but lack much mathematical theoretical foundation. This means it's challenging to reason about obvious ways to improve the performance of the networks, or how exactly the network learns efficiently.

Among some of the work that has looked at the deeper foundation for deep neural networks is Montufar et al. (2014), which showed that deep ReL networks capture exponentially more representation than a shallow network with the same amount of parameters, which suggest that depth, or hierarchies of representation, indeed is important. Similar results has been shown for a convex layered kernel model by Aslan et al. (2014), which suggests that layered representations are useful in general, and not only in neural networks.

The most surprising aspect of deep learning models is that despite a huge number of parameters to learn, and a loss surface that is expected to be highly irregular and non-convex, simple stochastic gradient descent manages to recover local minima of high quality. Some parallels, exact and approximate, have been drawn to models in statistical physics such as *variational renormalization* (Mehta and Schwab, 2014), and using connections to *spin-glass models*, Choromanska et al. (2014) recently showed that with some assumptions and a large enough network, the local minima tends to be concentrated in a band very close to the global minima. The assumptions that Choromanska used does not hold for deep neural networks, but she conjectured that local minima of these also tend to be concentrated in such a band. Furthermore, empirical studies by Goodfellow and Vinyals (2014) have shown that the loss surface usually is close to convex, which suggests that deep neural networks (or rather, layered representations) have specific characteristics that enable efficient gradient descent.

3.2.11 Technical details

When training our model, we used the architectures we described in chapter 3.2.9, which are known to work very well for general image classification tasks. Both of the built models were trained using Caffe[3], a framework for training convolutional nets, on GPU instances available in Amazon EC2. Instead of training the models from scratch, we used models that were already trained on the ILSVRC imagenet dataset, and then finetuned the model on our dataset. Most of the objects in the ILSVRC dataset are similar to objects that could be found in classified ads on *finn.no*, so the learned features should transfer well to our task. In Yosinski et al. (2014) it was shown that this kind of fine-tuning works well for tasks that are related, and even seems to improve generalization over training solely on the

[3] http://caffe.berkeleyvision.org/

Model	Accuracy
AlexNet	0.475
GoogLeNet	0.583

Table 3.2.1: Classification accuracy for our two architectures

training set. For fine-tuning, we fix the parameters of the model and only train the final logistic layer for 200000 iterations, when the loss on a separate testset seems to stop improving. We then unfix the parameters of the model and continue training for 100000 iterations to fine-tune the features. This way of fine-tuning has been found to work better than fine-tuning all the parameters in the whole model together in one go, see Branson et al. (2014).

As a training set from *finn.no*, we used approximately 1.3 million images, and to evaluate the error, we used a test set of 70 000 images. We ensured that the training set had an even distribution of classes, by using around 5000 images from each class. For some classes, where there were very few available images (less than 3000 images), we included the same images several times to ensure the classifier would not be biased away from these classes. Table 3.2.1 shows the classification error of our two trained models on the test set.

We also evaluated the confusion between the classes, plotted in figure 3.2.17. Several of the classes are ambiguous or overlapping and it's therefore not so surprising that some classes have more confusion than others. We also repeat that minimal cleanup was done to the image dataset, so there are several images which are misclassified, thereby reducing the actual score. The most surprising observation we made is that the classifier to some extent manages to separate male clothing from female clothing, something we originally did not think would be possible.

In figure 3.2.18 we show some examples of classifications from our trained classifier. We especially note that the classifier manages to correctly classify the jeans as women's jeans, presumably based on shape and style. The classifier also manages to choose the correct object to focus on in images with other objects, such as the image of an rc-plane, with a sofa in the background. In most cases the classifications below the top one are feasible classifications, but we note that in one of the images the classifier has given a relatively large probability to the class "tent"

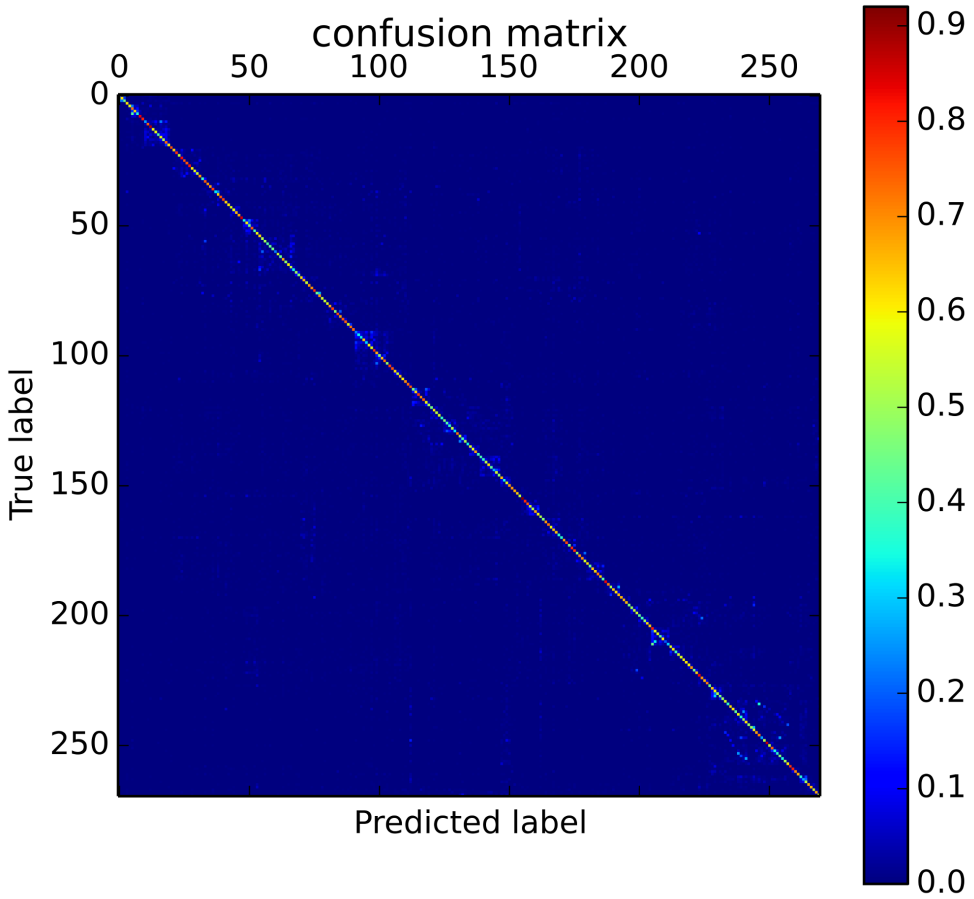


Figure 3.2.17: Confusion matrix for image classifier. Each coordinate represents the share of ads from category y that were actually classified as category x, encoded according to the color-scale on the right. For the category mappings, see appendix 7.1

which is not in the image.

We also show some images which were incorrectly classified in figure 3.2.19. Often misclassifications are due to "detail shots" of objects, such as the ones we show, which lack enough details to make out the main object, and thus confuses the classifier. Images with wrong orientation also often confuse the classifier.

Top 5 class probabilities :

Foreldre og barn/Barnemøbler/Senger : 0.78003
Møbler og interiør/Senger og madrasser/Senger : 0.19209
Møbler og interiør/Senger og madrasser/Madrasser : 0.01104
Møbler og interiør/Garderobe og skap/Garderober : 0.00384
Foreldre og barn/Barnemøbler/Stellebord : 0.00299

Top 5 class probabilities :

Klær, kosmetikk og accessoirer/Dameklær/Bukser : 0.80089
Klær, kosmetikk og accessoirer/Herreklær/Bukser : 0.18645
Foreldre og barn/Barneklær og sko/Gutt : 0.00455
Klær, kosmetikk og accessoirer/Herreklær/Skjorter : 0.00147
Foreldre og barn/Barneklær og sko/Jente : 0.00128

Top 5 class probabilities :

Møbler og interiør/Bord og stoler/Skrivebord : 0.73126
Møbler og interiør/Hyller og kommoder/TV-møbler : 0.04197
Møbler og interiør/Hyller og kommoder/Kommoder : 0.04015
Møbler og interiør/Hyller og kommoder/Nattbord : 0.02670
Møbler og interiør/Hyller og kommoder/Skjenk : 0.02344

Top 5 class probabilities :

Møbler og interiør/Belysning/Stålamper : 0.84219
Sport og friluftsliv/Jakt, fiske og friluftsliv/Telt : 0.09548
Hage, oppussing og hus/Byggevarer og oppussing/Gulv : 0.00759
Hage, oppussing og hus/Garasje : 0.00552
Møbler og interiør/Belysning/Taklamper : 0.00392

Top 5 class probabilities :

Fritid, hobby og underholdning/RC-utstyr/Fly : 0.95820
Fritid, hobby og underholdning/RC-utstyr/Båt : 0.02401
Møbler og interiør/Sofaer og lenestoler/Hjørnesofaer : 0.00896
Møbler og interiør/Sofaer og lenestoler/Sovesofaer : 0.00203
Møbler og interiør/Sofaer og lenestoler/Sofagrupper : 0.00129

Figure 3.2.18: Image classification examples

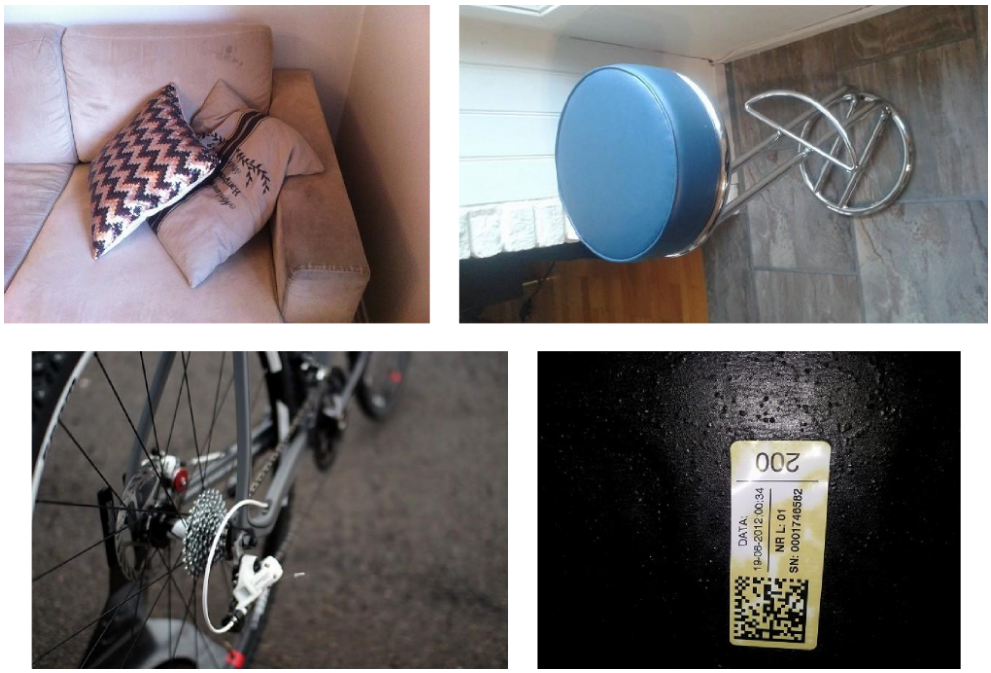


Figure 3.2.19: Examples of images that are wrongly classified, from top left : A 2-seater sofa (incorrectly classified as *cushions*), a chair (incorrectly classified as a *wall lamp*), a cyclocross bike (incorrectly classified as a *terrain bike*), and a dishwasher (incorrectly classified as *ice skates*)

3.3 Using raw features for image similarity

As an experiment we also tried out using the features in the layer before the final softmax layer as an embedding which could be used to compare images. While there is no theoretical foundation for this use of the features, several papers (Wan et al., 2014, Wang et al., 2014) report using the features in this way with relatively good results. Intuitively it makes sense that Euclidean distance based on the presence of features which are relevant for classification will make some sense as a similarity measure.

In figure 3.3.1 we did a 2D-embedding of the image features via a nonlinear embedding method called t-SNE. This embedding method seeks to find a two-dimensional embedding of our features that position similar items (in our case, according to the euclidean distance) close to each other and dissimilar items far apart, and thus retains as much of the similarity information in the 2D-embedding as possible. We refer to Van der Maaten and Hinton (2008) for more details. In the figure we can see clear clusters of for instance horses, cats, musical instruments and other objects. Interestingly, we also see that clusters of different animals are close to each other, as are different types of clothing. This illustrates that the extracted features has retained some form of similarity between classes, presumably based on similarities in appearance, and thus carries some meaning as a similarity embedding beyond a simple one-hot encoding of classes.

In figure 3.3.2 we show examples of similar images found via using our extracted image features as an embedding and searching for most similar images according to Euclidean distance. We can see that overall this works pretty well, and the most similar images tend to be of the same type of furniture. We can however see that for the second example not all of the similar images are strictly of the same type of chair. This might be because the image feature embedding focuses on irrelevant details in the image, such as the type of flooring in the background.

We also evaluate using the image features as an embedding and score it on our evaluation set described earlier, see table 3.3.1 for results. Note that in cases where there were more than one image in an ad, we've used the average of all the image features.

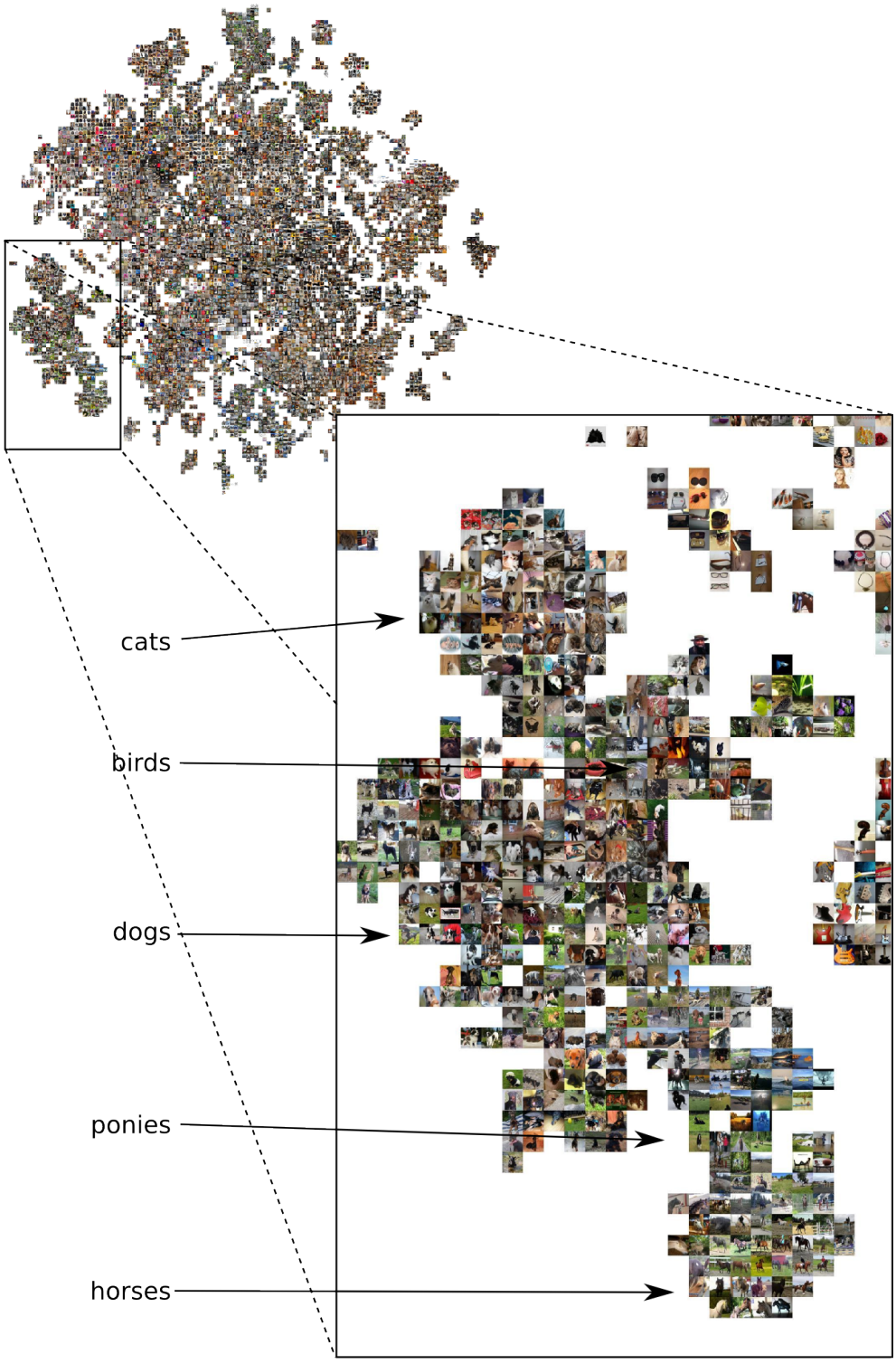


Figure 3.3.1: T-SNE 2D-embedding of the images from our *finn.no* classified ads, based on features from our deep neural network. The extract shows cluster of images with animals of different types.

Figure 3.3.2: Example of finding similar images via our image features embedding. Original image to left and top 5 most similar images in order of similarity to the right.

embedding	distance	coarsest score	coarse score	fine score
Image features	cosine	0.9430	0.7576	0.8280
Image features	euclid	0.8666	0.7007	0.8016

Table 3.3.1: Image similarity scored on our evaluation sets as described in chapter 2.3

4

Text analysis

In order to incorporate information from the text in our similarity measure, we need to find a suitable way to represent the text. In this chapter we will describe some common methods for representing text, how we may train a classifier on such representations, and evaluate using the text representations as embeddings for a similarity measure.

4.1 Representing text

To create a similarity metric based on the ad texts, we need some way to represent the text as features. A common method of doing this is to treat the occurrences of each specific word in a document as a value in a vector map, called a *bag-of-words* model. Given a fixed vocabulary of length N, we represent each document by a vector with length N, where each index in the vector specifies the number of times a specific word occurs in the document. As a simplified example, with a very short vocabulary of the words:

$$[\text{a}, \text{an}, \text{is}, \text{the}, \text{this}, \text{and}, \text{but}, \text{text}, \text{example}, \text{very}, \text{long}, \text{short}]$$

the text "This is an example text, but the text is not very long." will have the bag-of-words representation :

$$[0, 1, 2, 1, 1, 0, 1, 2, 1, 1, 1, 0]$$

A common modification to the bag-of-words model, is to weight each word count by its *inverse document frequency*, the logarithm of the number of documents in our dataset divided by the number of documents which contain the word:

$$\text{idf}(t, D) = \log\left(\frac{N}{n_d}\right) \tag{4.1}$$

This weighting is intended to weight down words which frequently occur in all documents, and weight up words which rarely occur in documents. The assumption is that frequently occurring words carry little discriminate information about the documents, and that rare words may be more important. The modification, which usually is called *tf-idf weighting*[1], is commonly used in information retrieval tasks, for e.g. finding similar documents in large datasets.

We will evaluate using the tf-idf weighted bag-of-words representations as an embedding for our similarity measure, but instead of the inverse document frequency as defined in (4.1), we will use a slightly different version which is used in a popular document similarity algorithm called *MoreLikeThis*:

$$\text{idf}_{mlt}(t, D) = 1 + \log\left(\frac{N}{1 + n_d}\right) \tag{4.2}$$

The *MoreLikeThis* algorithm embeds documents using the tf-idf weighted bag-of-words model with inverse document frequence defined as in (4.2), and finds documents similar to a specific document by measuring the cosine distance between their embeddings[2]. Since this document similarity algorithm is common to use for e.g. recommending documents, it serves as a useful baseline for our ad similarity measures. For the results of using this similarity measure on our

[1] since the entries in the bag-of-words representation are the *term frequencies* multiplied by the *inverse document frequencies*

[2] with some slight modifications we won't go into here

evaluation set, see table 4.4.1 later this chapter.

A drawback with both the bag-of-words representations and the tf-idf weighted representations, is that the feature vectors will have length equal to number of unique words in the entire corpus (unless we decide to ignore some words). Since there are a large number of unique words in our ad corpus, the feature vector for each ad will be very sparse and high-dimensional, which a lot of classifier methods will not handle well. These representations also have the shortcoming that they don't take into account that different words might have similar meaning, such as for instance "car" and "automobile", "big" and "large", etc.

A method for representation of texts which handles this shortcoming better, are *topic models*. These models can represent texts by a vector of latent topics which summarize the contents of the text. Since this vector is significantly shorter than the bag-of-words representation, it is often a better choice of representation for many tasks.

A topic consists of clusters of words that frequently occur together, such as for instance the topic "clothing", which might contain the words "shirt", "pants", "jacket", "sweater", and so on. Typically the number of possible topics are fixed at the outset, and the topics are learned in an unsupervised manner over a corpus of texts relevant to the task at hand. We will describe the topic model we will use, *latent dirichlet allocation*, in the next part.

4.2 Latent Dirichlet Allocation

Latent Dirichlet Allocation (Blei et al., 2003) is a graphical model for topic discovery. The assumptions of the model are that the texts were created by a probabilistic generative process with some unknown *latent* parameters.

More formally, the model assumes that we have K topics, and each text d_i consists of a mixture of these topics, θ^{d_i}, where θ^{d_i} has been sampled from an overall dirichlet prior with parameter α. Furthermore, each topic has an associated categorical word-distribution parametrized by φ_k, where φ_k has been sampled from another overall dirichlet prior with parameter β. The model then assumes that each word $w_i^{d_i}$ in the texts were generated by first sampling a topic $z_i^{d_i}$ from the text-specific (categorically distributed) topic mixture and then sampling a word

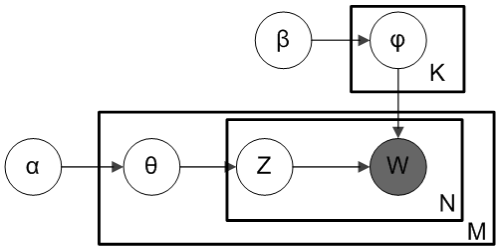


Figure 4.2.1: Plate model of Latent Dirichlet Allocation

from the topic-specific word distribution. Overall the model is defined as:

$$\begin{aligned} w_i|z_i, \varphi_k &\sim \text{Cat}(\varphi_k) \\ \varphi_k &\sim \text{Dirichlet}(\beta) \\ z_i|\theta_i &\sim \text{Cat}(\theta_i) \\ \theta_i &\sim \text{Dirichlet}(\alpha) \end{aligned} \tag{4.3}$$

where K, α and β are hyperparameters of the model. The model is also often represented by a plate model such as the one in figure 4.2.1 where M is the number of texts, and N is the number of words in a given text. We note here that the choice of a dirichlet distribution as prior stems from the fact that the dirichlet distribution is the conjugate prior of the categorical distribution.

Learning the topics distribution of any text in the corpus is thus a problem of learning the most likely parameters θ,φ,$\mathbf{z}$ of the LDA model. This would usually be done by calculating the posterior probability:

$$p(\theta, \varphi, z|w, \theta, \beta) = \frac{p(\theta, \varphi, z, w|\alpha, \beta)}{p(w|\alpha, \beta)}$$

Unfortunately, this distribution is intractable[3] to compute due to $p(w|\alpha, \beta)$, so we have to resort to approximate inference instead of exact methods. A number of approximate methods are available, such as variational inference (Blei et al., 2003), expectation propagation (Minka and Lafferty, 2002), or Markov Chain Monte Carlo with gibbs sampling (Griffiths and Steyvers, 2004). In the next section we will describe inference in the Latent Dirichlet Allocation model with collapsed

[3]the intractability stems from the fact that there is no analytical solution to the integral we have to solve to calculate this probability, and that numerical integration in high dimensions is in general intractable. For details, see part 5.1 in the original LDA paper

Gibbs sampling.

4.2.1 Inference with (collapsed) Gibbs Sampling in LDA

The original LDA paper by Blei et al. suggests using variational inference to estimate parameters θ,φ,z. As a faster alternative, Griffiths and Steyvers (2004) suggested to instead sample from the posterior $p(z|w)$, then use this sample to estimate the statistics θ and φ. For sampling from the posterior $p(z|w)$ we can use MCMC methods such as gibbs sampling, which is simple to implement and reasonably fast.

They show that it's possible to express the posterior for an individual topic assignment conditional on all other topic assignments as:

$$p(z_i = j|\mathbf{z}_{-\mathbf{i}}, \mathbf{w}) \propto \frac{n_{-i,j}^{(w_i)} + \beta_j}{n_{-i,j}^{(\cdot)} + W\beta_j} \frac{n_{-i,j}^{(d_i)} + \alpha_j}{n_{-i,\cdot}^{(d_i)} + K\alpha_j}$$

where z_i is the topic assignment of the current word, $\mathbf{z}_{-\mathbf{i}}$ are all the topic assignments of all the words in the texts *except* z_i, $n_{-i,j}^{(w_i)}$ is the number of times the word w_i has been assigned to topic j (excluding the assignment of the current word), $n_{-i,j}^{(w_i)}$ is the number of times any word in document d_i has been assigned to topic j (excluding the assignment of the current word), and W is the number of unique words in the texts. Note that this expression does not depend on θ and φ, since it's possible to integrate out these parameters (see Griffiths (2002) for details).

Thus for our gibbs sampling method we initially randomly assign topics to each word, and then iterate over the entire text, sampling from the posterior for each word individually conditional on the other parameters $\mathbf{z}_{-\mathbf{i}}$. We do several passes over all the texts as *burn-in*, and then choose a final sample at a fitting number of passes. At any step we can then estimate θ and φ from our sample via the estimates:

$$\varphi_j^{(w)} = \frac{n_j^{(w)} + \beta_j}{n_j^{(\cdot)} + W\beta_j} \tag{4.4}$$

$$\theta_j^{(d)} = \frac{n_j^{(d)} + \alpha_j}{n_\cdot^{(d)} + K\alpha_j} \tag{4.5}$$

Note that once the parameters of the model are learned, we can also extract the topic-parameter θ for new texts that are not part of the original corpus, via doing gibbs sampling on a new document and extracting the parameters θ_d for that document based on the same variables.

In early LDA implementations, it's been common to set the hyperparameters α, β as fixed symmetric priors, i.e. $\alpha_1 = \alpha_2 = \cdots = \alpha_n$, since this simplifies inference. In the case of β, the prior over the word-distributions, this seems feasible, since this means we assume any word is equally likely for a specific topic. In the case of α, the prior over the topic distributions, this might however seem more problematic, since few text collections would be expected to have a symmetric distribution of topics. Instead, they rather usually have some dominating topics and multiple smaller related topics. Forcing the topic-distributions to be symmetric might lead some topics to include words that may not make sense, just to ensure an equal distribution of topics. Confirming this intuition, Wallach et al. (2009) showed that using asymmetric priors for α has a strong impact on the quality of the inferred topics, and therefore suggest that correctly specifying the hyperparameter α is very important.

The wholly bayesian way would be to infer α together with z using e.g. hierarchical bayesian inference, but as Wallach showed, a much faster and equally precise method is to simply initially fix the hyperparameter as a symmetric prior, and then later estimate the hyperparameter independently during the gibbs sampling. We therefore use this approach and optimize the hyperparameter α every 10 epochs after the burn-in period is done. An iterative method for estimating dirichlet parameters from count data can be found in e.g. Minka (2000).

4.2.2 Technical Details

For our inference, we used the software package MALLET[4], which was developed at the University of Massachusetts Amherst, and is a well tested and optimized package for LDA via collapsed Gibbs sampling. We use as input the text from a sample of 300 000 classified ads in our dataset. As preprocessing we first concatenate the header of the ad and the body text, remove any non-alphanumeric characters, and remove some common stopwords that has little value for our task (see details in appendix 7.3) and then tokenize. We ran the gibbs sampling

[4] http://mallet.cs.umass.edu/

procedure for 1000 passes over the corpus, since the log-likelihood seemed to plateau after that.

We also tried out varying the preset number of topics, from 1000 to 2000, and evaluated how many to use by inspecting classifier accuracy on a held out test set of 20000 ads. See table 4.2.1 for the classifier results. For the final classifier we settled on 1500 topics since quality on our classification task did not improve by adding more topics than this.

Table 4.2.1: Classifier accuracy for choice of number of topics

Number of topics	Classifier accuracy
1000	0.662
1500	0.690
2000	0.686

In table 4.2.2 we see the top words for some of the resulting topics. Many closely correspond to different categories in the *finn.no* dataset, or even subsets of categories, while others simply represent qualitative words that are not of high value for our task. Overall LDA manages to separate very well plausible topics in our ad corpus.

4.3 Ad classification

As part of our multi-modal semantic embedding which we will describe in chapter 5, we wish to train an ad classifier based on the text in the ads. We will use the learned per-document topic probabilities from the latent dirichlet allocation model as *features* for such a classifier. The output of the topic model for a specific ad will be a vector of 1500 probabilities which are the parameters to the inferred multinomial distribution of topics for that ad.

For our classifier we could choose from a number of methods, but the size of the training set limits the classifiers which are practical to use. Initially we tried out *random forests* (Breiman, 2001), an ensemble classifier that has been shown to perform well on a number of varying tasks, but since training this classifier requires to keep the entire training set in memory, training and optimizing parameters took much too long time. Instead we opted for logistic regression fitted with stochastic

Table 4.2.2: A selection of learned topics from our model

topic nr	top words with probability > 0.01	possible topic
0	hundebur buret herkules bur passer mål bil hund sammenleggbart solid bilbur stort hunder	hundebur
1	cm ca høyden lengden bredden måler justeres ned dybden toppen	mål
4	ski cm str binding scarpa skiene bindinger telemark støvler rottefella ntn telemarkski marker nye lengde mm crispi feller	ski
5	glitter neglelakk shades nail uv neglelakker essie coat deborah lippmann opi fifty pink negler cnd top forskjellige lakker base grey skimmer start gelenegler manikyr gel negl gele shellac nyx gellack	neglelakk
7	kr sekker sekk liter tørr bjørkeved bjørk pr blandingsved veden liters salgs fin furu gran cm bestilling	sekker med ved
9	godt meget god passer både gode svært egner veldig bra samt alle ypperlig kvalitet fungerer egnet fleste vår	kvalitative (positive) ord
11	undertøy korsett sexy truse størrelse korsettet size sort filippa nattkjole spandex materialet festes nylon blonder farge bh snøring nattkjoler stil hudfarget korsetter salg strømper råsexy	undertøy
13	iphone gb telefonen lader følger hvit alle ulåst kvittering eske telefon skjerm strøken apple åpen deksel tlf fungerer glass	iphone
14	db hz mm khz power output ohm frequency ohms input kg line impedance stereo weight response dimensions mv watts	måleenheter for lydutstyr
15	pedal marshall boss bass pedaler kabinett overdrive amp forsterker ampeg gitar strøken distortion chorus effekter	elgitarutstyr
16	pc asus gaming gtx gb pcen tastatur geforce nvidia spill mus ssd skjerm kraftig windows ultra logitech cpu hdd kjører full	pc
19	peis jøtul peisinnsats dovre peisen innsats demontert klar nordpeis hentes fin henting ovn stein komplett omramming	peis
22	stål rustfritt børstet aluminium ny design solid farge laget metall lekker tre front glass rustfri passer	material-beskrivelser

gradient descent, since this allows us to train our model using only a single training sample at a time. When comparing the classifiers on a smaller training set, we also found that logistic regression seemed to reach equivalent accuracy, presumably because a linear model is sufficient to separate the classes in our data set. In the next section we describe how we trained our logistic regression model.

4.3.1 Logistic regression

There exist a number of methods to fit logistic regression models, but we choose to use stochastic gradient descent, since this allows us to avoid loading the entire dataset in memory. As in the image analysis case, we seek to optimize log-likelihood with additional L2-regularization to avoid overfitting. The loss function we use in the gradient descent is thus:

$$L(\beta) = -\frac{1}{N}\sum_{i=1}^{N}\log(\mathrm{p}(y_i = k_i|\beta)) + \alpha \cdot \frac{1}{2}\sum_{b=0}^{1500}\sum_{m=1}^{K}\beta_{b,m}^2$$

where k_i is the true class of y_i and α is a parameter for the strength of L2-regularization. Similar to before, we use the iterative update scheme:

$$\beta_{t+1} = \beta_t - \eta \cdot \nabla L(\beta_t)$$

where η is the learning rate. We use an adaptive learning rate which gives us gradual reduction in learning rate and ensures convergence.

4.3.2 Technical details

For fitting the multinomial logistic regression with stochastic gradient descent, we use the python implementation found in Scikit-learn[5]. Prior to training our classifier, we normalize the topic probabilities to have zero mean and unit standard deviance, since stochastic gradient descent tends to be sensitive to the scaling of the variables. In order to choose the α parameter for regularization, we use cross-validation and randomly choose a training set of 300 000 ads and a test set of 20 000 ads from our complete set of around 3.5 million ads. Note that to ensure a balanced training set, we tried to choose around 1200 examples from each class. Unfortunately some categories had less than 1200 examples in the entire dataset, so

[5] http://scikit-learn.org/

we weighted up these class examples in the loss function to ensure an unbiased classifier. For the results of our evaluation of the α parameter, see table 4.3.1.

Table 4.3.1: Classifier accuracy for choice of α

α	Accuracy
0.01	0.614
0.001	0.671
0.0001	0.683
0.00001	0.689
0.000001	0.667

Since accuracy seems to decrease for values of α above 0.00001, we fix the α of our model at $\alpha = 0.00001$. Our final accuracy on a separate test set is 0.686.

To investigate further which classes are well predicted and which are not, we generate a confusion matrix (see figure 4.3.1). Most of the classification errors are due to confusion between similar categories, such as different races of cats and dogs, various types of sofas ("2-seater", "3-seater", "sofa group", "corner sofa"), different types of books ("fiction", "cookbooks", "non-fiction", "encyclopedias", "other"), male and female clothing, etc. This is not so surprising, since many of the ads are hard to place in a correct category even for a human being just based on text. Additionally many of the categories are partially overlapping (such as "food processor","blender" and "mixer") which means it may be ambiguous which category is correct. Investigating examples of erroneously classified ads, we also find many ads that are wrongly categorized by the users, see table 4.3.2 for some examples. As we noted in the introduction, we have not attempted to clean these categorizations, so this has a significant impact on our classifier accuracy. Based on a small sample of ads, we've estimated that the proportion of ads that are either wrongly categorized or contain several items of multiple categories are around 11%, so the quality of classifier could probably be 11-12% more precise if we disregarded these erroneous ads. Finally, we note that for a small amount of ads there is not enough descriptive text to be able to tell which category the ad belongs to, see table 4.3.3 for some examples.

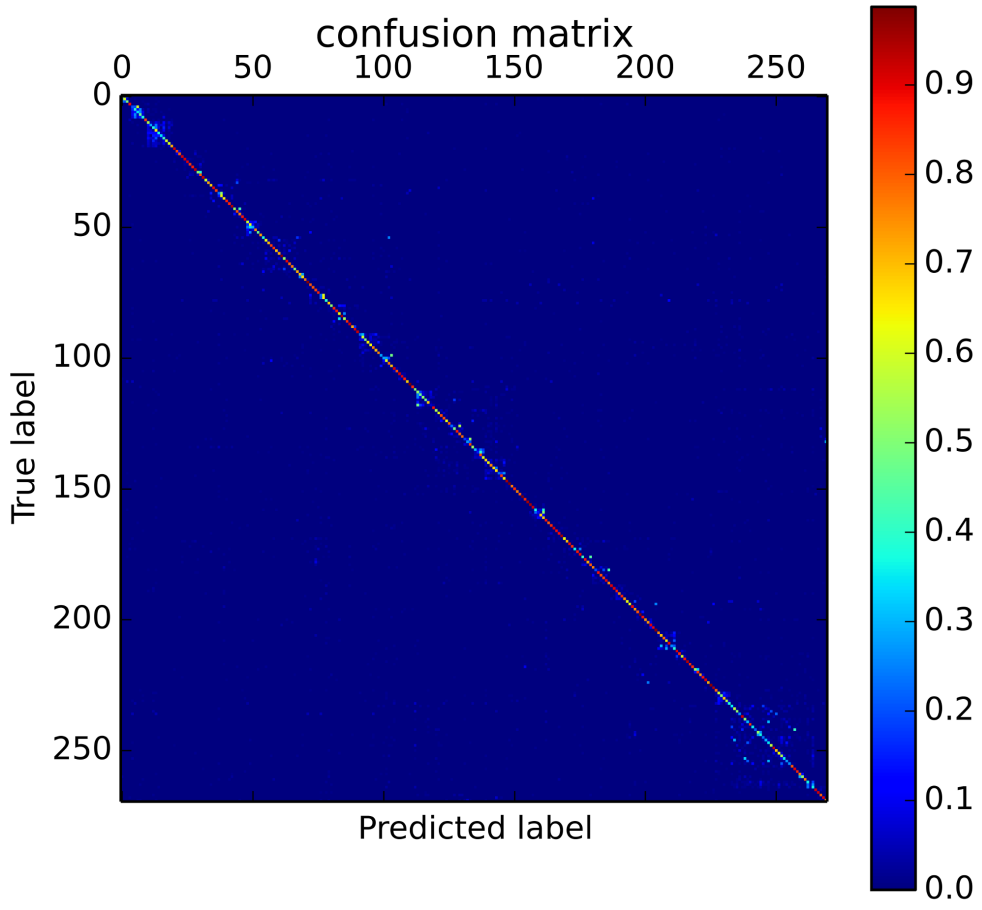


Figure 4.3.1: Text categorization confusion matrix. Each coordinate represents the share of ads from category y that were actually classified as category x, encoded according to the color-scale on the right. For the category mappings, see appendix 7.1

Category	Ad heading
.../Musikkinstrumenter/Keyboard/synth	Cello selges
.../Ballsport/Basketball	Ståbrett til barnevogn
.../Herreklær/Skjorter	Ralph Lauren dameskjorte
.../Verksted, bygg og anlegg/Byggtørkere	oljefat eller bensinfat ønskes
.../Foto og video/Kompaktkameraer	NIKON SPEILREFLEKSKAMERA

Table 4.3.2: Examples of ads that are wrongly categorized

Ad heading	Ad body text
Gis bort	Gis bort
Mange fine klær.	Kjøper betaler frakt.
alt selges sammen	Alt befinner seg i Oslo
Gis Bort!	Send sms eller epost hvis interessert.
Gudsjammerlig støgt ræl fra dypet av garasjen	Far rydder, alt må vekk.

Table 4.3.3: Examples of ads that are hard to classify based on text alone

4.4 Evaluation of text-based similarity measures

As a baseline for comparison, we evaluate using the idf-weighted bag-of-words representations as embeddings for our similarity measure. We also evaluate using the topic features learned from our latent dirichlet allocation model as embeddings. The results can be found in table 4.4.1 below.

embedding	distance	coarsest score	coarse score	fine score
Tfidf-weighted bow	cosine	0.6807	0.5985	0.7360
Text topics	cosine	0.7930	0.7209	0.7992
Text topics	euclid	0.7191	0.6888	0.7810

Table 4.4.1: Text similarity scored on our evaluation sets as described in chapter 2.3

We see that the tf-idf weighted bag-of-words embeddings work poorer than our topic-based embeddings. This is probably because there are few relevant words in common between the ad examples, and so the cosine distance may be a bit arbitrary for the negative and positive examples, i.e. the measure is not able to rank which example is most similar. This somewhat validates our intuition that there is value in focusing on the topic contents of our ads rather than the individual words.

You shall know a word by the company it keeps.

J.R. Firth

5

Multi-modal analysis

The main idea of this thesis is to be able to use information from both text and images to compare ads. One way to do this is to induce some embedding $f(txt, im)$ to compare ads so that e.g. $||f(txt_1, im_1) - f(txt_2, im_2)||$ is small for similar ads, and large for dissimilar ads. The rigorous way to do this is to jointly model the embedding function for both images and text, either generatively or discriminatively. Doing this discriminatively does however require ground truth information on the similarity of the ads, which may not be available, and doing it generatively requires being able to define a feasible generative model, which is not easy, especially for images, and may also require more data to train than we have available. Instead, we will explore two simpler methods which does not involve joint modeling.

While both of these techniques are expected to perform poorer than joint modeling of images and texts, they are simple techniques that only rely on class data for the ads and (in the case of the second model) a large corpus of text, both of which are easily available in our case.

5.1 Method 1 : Concatenating features

The first "naive" method we test out is simply based on *concatenating* the topic probabilities from our topic model with the features from our image model, and using these directly as an embedding. Thus the complete feature-vector is:

$$f_{conc} = f_{topic} || f_{imfeat} = (x_1, x_2, \ldots, x_{1500}, y_1, y_2, \ldots, y_{1024})$$

Using the concatenated features as an embedding does not have a ready probabilistic interpretation, but has been used in e.g. Kiela and Bottou (2014) with satisfactory results.

Since the topic probabilities are bounded in $[0, 1]$ while the image features may be much larger, we normalize the features before concatenating them to get them on approximately the same scale. So our concatenated feature vector thus becomes:

$$f_{conc} = \frac{f_{topic}}{||f_{topic}||} || \frac{f_{imfeat}}{||f_{imfeat}||} = (\frac{x_1}{||x||}, \frac{x_2}{||x||}, \ldots, \frac{x_1}{||x||}, \frac{y_1}{||y||}, \frac{y_2}{||y||}, \ldots, \frac{y_{1024}}{||y||}) \quad (5.1)$$

5.2 Method 2 : Semantic embeddings

Our second method relies on using information about the similarity of objects from a separately learned embedding. The technique which we will use is similar to the one described for images by Norouzi et al. (2013), and relies on projecting the classification probabilities from our text and image classifiers into a "semantic" vector space induced by a learned word-embedding. In the next section we will describe "semantic" word-embeddings, and how they can be learned from a large text set.

5.2.1 Semantic embeddings

A word-embedding is a function $f(\cdot)$ so that each word is represented by a n-dimensional real-valued dense vector, or mathematically:

$$f(w) : \mathbb{R}^m \rightarrow \mathbb{R}^n$$

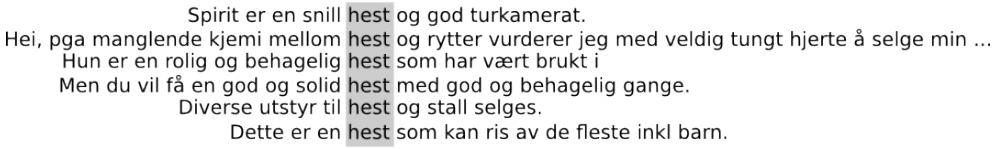


Figure 5.2.1: Example of word context

where w is an m-dimensional one-hot encoding representation of the word, i.e. $w = (0, 0, \ldots, 0, 1, 0, \ldots, 0, 0)$, and usually $n \ll m$.

We want to learn a word-embedding that manages to embed words so that words with similar meanings also are close to each other in the vector space formed by the embedding. Since such a constraint implicitly means we are conserving the "meaning" of the words, such an embedding is often called a "semantic" embedding, and the vector space induced by the embeddings is called a "semantic" vector-space. Learning such an embedding usually relies on what is called the *distributional hypothesis*, namely the hypothesis that words that occur in similar contexts have similar meanings.

To illustrate this hypothesis, we can use an example sentence, say "I'm driving my X on the highway", where X is a word with an unknown meaning. From interpretation of this sentence, it's easy to see that the set of objects that can be *driven*, and which belong on the highway, are objects such as *vehicles*, so we can deduce that X means some sort of vehicle. However, we do not strictly need to interpret the sentence, as purely from empirical data we can see that words that usually occur in such a context are words such as "car","sportscar","trailer" or even "bicycle", while words such as "orange", "hammer", or "yellow" are highly unlikely, and thus the word X probably is in some sense similar to the words "car", "sportscar" etc., and in some sense dissimilar to the words "orange","hammer" and "yellow". In other words, it is feasible to deduce the semantic similarity of words based on purely empirical distributional data, given that we have enough of it.

Based on the distributional hypothesis, a simple way to create a word-embedding that achieves our goal, is to directly represent words by how often they appear in specific contexts, for instance through a co-occurrence matrix. (See table 5.2.1 for a simplified example of such a co-occurrence matrix). We can limit the context to mean the words in the immediate neighborhood of the specific word, for instance a window of the five words before and after the current word. Now, if the

distributional hypothesis is correct, and we have enough data, then similar words will appear in similar contexts, meaning the distance between the corresponding vectors formed by the rows of the co-occurrence matrix will be relatively small.

Table 5.2.1: Example of co-occurrence matrix for sentences "I like deep learning.", "I like NLP." and "I enjoy flying."

counts	I	like	enjoy	deep	learning	NLP	flying	.
I	0	2	1	0	0	0	0	0
like	2	0	0	1	0	1	0	0
enjoy	1	0	0	0	0	0	1	0
deep	0	1	0	0	1	0	0	0
learning	0	0	0	1	0	0	0	1
NLP	0	1	0	0	0	0	0	1
flying	0	0	1	0	0	0	0	1
.	0	0	0	0	1	1	1	0

There is a problem with this approach : the co-occurrence matrix will be of dimensions $N \times N$, where N is the size of the vocabulary. Since for most cases, N will be large, the matrix will be extremely high-dimensional, which makes it infeasible to work with in most applications. A solution to this could be to do some kind of dimensionality reduction in order to reduce the matrix to feasible dimensions, for instance via rank-reduction with SVD (Singular Value Decomposition) as is done in Deerwester et al. (1990).

While it would be possible to use regular SVD for dimensionality reduction, it is unfortunately computationally expensive to calculate the SVD for such a large matrix. Instead we're going to use a method that achieves approximately the same goal, though in a roundabout way. This method, called the *skip-gram model with negative sampling* and introduced by Mikolov et al. (2013), is significantly faster and allows us to train our embedding on a very large body of text in a matter of hours. This method is originally based on a quite different approach, trying to cast the embedding task as a supervised prediction problem where we implicitly learn the embedding. The original "skip-gram model" tries to find a word-embedding that is useful for predicting context words for a specific word. This is done by optimizing

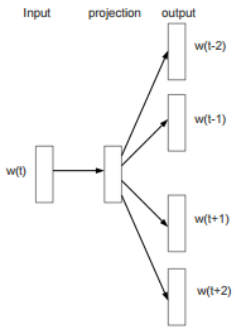


Figure 5.2.2: Simplified figure of skip-gram model

the log-probability :

$$\frac{1}{T}\sum_{t=1}^{T}\sum_{-s\leq j\leq s, j\neq 0}\log p(w_{t+j}|w_t) \tag{5.2}$$

where $w_1, w_2, \ldots, w_T$ is a sequence of words in a sentence, s is the size of the context window, and $p(\cdot|\cdot)$ is defined as

$$p(w_O|w_I) = \frac{\exp(f(w_O)\cdot f(w_I)^T)}{\sum_{w\in V_w}\exp(f(w)\cdot f(w_I)^T)}$$

where V_w is the set of all words in our text dataset, and $f(\cdot)$ is the word-embedding, which is the parameter to be learned.

In order to improve the training efficiency, Mikolov et al. (2013) suggested the skip-gram model with *negative sampling*, a modification based on *Noise-Contrastive Estimation* (Gutmann and Hyvärinen, 2010). Given a word w and its set of context words c, we can model the probability that a given word-context pair (w, c) came from our dataset with the simple model:

$$\begin{aligned} p(D=1|w,c) &= \sigma(f(w)\cdot g(c)) = \frac{1}{1+\exp(-f(w)\cdot g(c))} \\ p(D=0|w,c) &= 1 - p(D=1|w,c) \end{aligned} \tag{5.3}$$

where $p(D=1|w,c)$ is the probability that the word-context pair (w,c) came from our dataset and $p(D=0|w,c)$ is the probability that the word-context pair (w,c) did *not* come from our dataset, $f(w)$ is a word-embedding and $g(c)$ is a context-embedding, both of which are parameters to be learned. These

embeddings are learned via maximizing $p(D = 1|w, c)$ for observed (w, c) pairs and maximizing $p(D = 0|w, c)$ for randomly sampled *unobserved* (i.e. negative) (w, c) pairs. The unobserved pairs are created via sampling the context c from the empirical distribution over the dataset, i.e.

$$p_N(c) = \frac{\#c}{\sum_{c \in V_c} \#c}$$

where V_c is the set of all contexts in our data set.

Hence, instead of optimizing the log-probability in (5.2), the negative-sampling skip-gram model optimizes this objective :

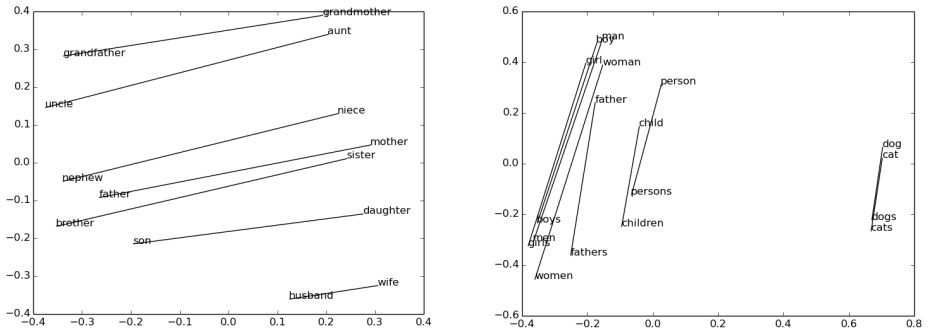
$$\sum_{w \in V_w} \sum_{c \in V_c} \#(w, c)(\log \sigma(w \cdot c) + k \cdot E_{c_N \sim p_N}[\log \sigma(-w \cdot c_N)]) \tag{5.4}$$

where $\#(w, c)$ is the number of occurrences of the word-context pair (w, c) in the dataset and k, a hyperparameter of the model, is the number of negative samples per word-context pair. This objective maximizes the dot-product $f(w) \cdot g(c)$ of frequently occurring word-context pairs, and minimizes it for random *unobserved* word-context pairs, thus observed word-context embeddings will be close to each other (in terms of cosine distance, see (2.4)), and unobserved word-context embeddings will be far from each other. So if two similar words frequently occur in the same context, their word-embeddings will both be close to the embedding of this context and thus close to each other.

The objective in (5.4) is an approximation to the objective in (5.2), but is significantly faster to train, and has been shown to perform surprisingly well for word-embedding tasks such as ours. The reason that this performs so well, was shown by Levy and Goldberg (2014) to be because optimizing this objective is equivalent to factorizing a co-occurrence matrix as described earlier, with shifted Pointwise Mutual Information as entries:

$$M_{ij} = W_i \cdot C_j = w_i \cdot c_j = \text{pmi}(w_i, c_j) - \log k$$
$$\text{pmi}(x, y) = \log \frac{p(x, y)}{p(x)\, p(y)}$$

Thus the word-embeddings produced by the skip-gram model with negative



(a) Male-Female words **(b)** Singular-Plural words

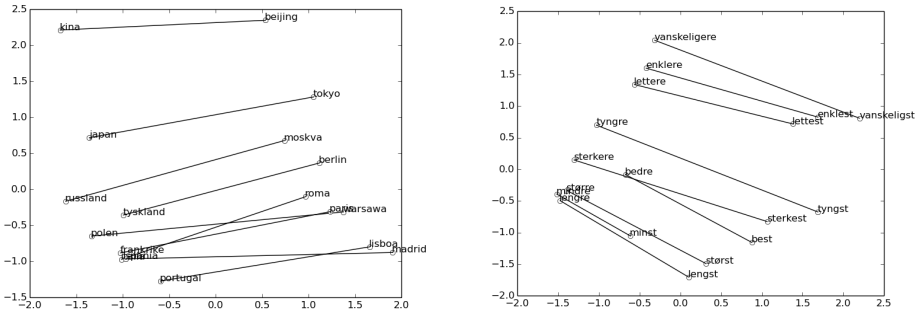
Figure 5.2.3: Examples of embeddings from an english word-embedding model (dimensions reduced with PCA)

sampling comes from the factorized part of a word/context co-occurrence matrix. Note that the learned context-embeddings C_j are not used.

The embeddings induced by this model has been shown to do a very good job of conserving semantic relations and even allows us to use a simple form of arithmetic to combine words. A much-cited example of this is where Mikolov et al. (2013) show that the embedding of "queen" is close (in terms of cosine distance) to the embedding of "king" minus "male" plus "female" :

$$f(\text{king}) - f(\text{male}) + f(\text{female}) \approx f(\text{queen})$$

This might seem surprising, but the likely reason is that the model enforces several types of similarity for each word - for instance the term "king" is in some sense similar to other royal terms, but it is in some sense also similar to other male "occupations", while the term "queen" is in some sense similar to female occupations. Seemingly, the model automatically learns and embeds many such types of similarities along different axes of the vector-space. An illustration of this can be shown by using PCA to project the word-embeddings in two dimensions, see figure 5.2.3.



(a) Capitals **(b)** Superlatives

Figure 5.2.4: Some of our learned embeddings (dimensions reduced with PCA)

5.2.2 Technical details

For our task, we use the toolkit *word2vec*[1] which was made public by Google at the same time as the paper describing the negative sampling skip-gram model. Our model is trained part on text from *Norsk Aviskorpus*[2], a set of news articles from 10 norwegian newspapers from 1998 to 2011, a text dump of the contents of norwegian Wikipedia, and text from *finn.no* classified ads. Altogether we have a body of text consisting of 817.4 million words, where 72.4% (591.7 M) is from the news articles, 10.2% (83.0 M) is from wikipedia, and 17.4% (142.7 M) is from the classified ads. Prior to feeding our text to the model, we clean out all non-alphabetic characters, and convert all uppercase characters to lowercase.

The negative-sampling skip-gram model includes a number of parameters that needs to be specified. We follow the recommendations published by Levy et al. (2015) and use a context window of size 10, negative-sampling of size 10, and word-embedding vectors of length 300. Overall, the fitting took around 9-10 hours on a quad-core laptop computer.

See figure 5.2.4 for some examples of embeddings from the trained model.

5.2.3 Projection into semantic vector space

To enable multi-modal comparison, we follow the method suggested by Norouzi et al. (2013). Given the class probabilities for our text- and image-classifier, denoted by $p_{im}(ad_n = cl|im_n)$ and $p_{txt}(ad_n = cl|txt_n)$ respectively, where ad_n is a

[1] https://code.google.com/p/word2vec/
[2] http://avis.uib.no/avis

classified ad with associated image im_n and text txt_n, and cl is a category from the set C of all *finn.no* categories, we propose to find an embedding of the ad in the semantic vector-space by combining the word-embeddings of the classes via using the probabilities from the classifiers. More specifically:

$$f(ad_n) = \frac{1}{\#C} \sum_{cl \in C} [w_{im} \cdot p_{im}(ad_n = cl|im_n)s(cl) + w_{txt} \cdot p_{txt}(ad_n = cl|txt_n)s(cl)]$$

where $s(\cdot)$ is the word-embedding of the class cl, and w_{im}, w_{txt} are weights to be learned by e.g. cross-validation.

For the word-embeddings of the classes, we try to find the closest word which succinctly describes the class. If no such word exists, we average several words which are close. For an overview of the exact mappings between classes and words, see appendix 7.2.

While the ad consist of just one body text and header text, the number of images per ad may vary from none to over hundred. We evaluated taking the max of each class probability over the probability output for all images in an ad, but since this did not perform any better than simple averaging of probabilities when trying to classify images, we settled for simply averaging the probabilities of each image. Thus our final embedding is :

$$f(ad_n) = \frac{1}{\#C} \sum_{cl \in C} \left[w_{im} \cdot \left(\frac{1}{I} \sum_{i=1}^{I} p_{im}(ad_n = cl|im_{n,i})s(cl) \right) + w_{txt} \cdot p_{txt}(ad_n = cl|txt_n)s(cl) \right]$$

5.3 Evaluation of multi-modal similarity measures

We evaluate the multimodal similarity measures we described on our evaluation set, see results in table 5.3.1 below.

We see that the semantic embedding performs slightly better than the "naive" method based on concatenating features for the class based evaluation sets. However, for the fine-grained evaluation set, the semantic embedding performs very poorly. With some further investigation, we found that this was not so surprising, since whenever our classifier is correct and confident (which happens to

embedding	distance	coarsest score	coarse score	fine score
Conc. text/image feat.	cosine	0.9608	0.8076	0.8626
Conc. text/image feat.	euclid	0.9608	0.8076	0.8626
Semantic embedding	cosine	0.9801	0.8230	0.6704
Semantic embedding	euclid	0.9705	0.8219	0.6710

Table 5.3.1: Multi-modal similarity measures scored on our evaluation sets as described in chapter 2.3

be most of the time), objects from the same class will have the same semantic embedding, thus the difference between dissimilar objects and similar objects will be zero, and the model is not able to distinguish between objects. This limits the application of the semantic-embedding model for any fine-grained similarity tasks.

6

Conclusion and Summary

We have evaluated several similarity measures, for the task of comparing classified ads from *finn.no*. The similarity measures were based on extracting features from text and images in classified ads and combining them in some manner in order to use information from both text and images. The simplest of these methods were based on simply using the extracted features themselves as embeddings, and applying common distance metrics over the embeddings to measure similarity between ads. A more complex method was suggested, training separate text and image classifiers, and projecting text and image class probabilities into a separately learned "semantic" embedding. While there was some advantage in using this method for comparing class-based triplets, the method failed to be able to distinguish fine-grained triplets, and thus was a poor candidate for measuring similarity between ads. We did however find that our simpler method of concatenating features from a topic model and features from deep convolutional network, performed satisfactory as a similarity measure on all our evaluation sets, and better than any of the similarity measures using only text or image information.

It is thus clear that using information from both images and text gives a significant advantage over using information from either text or images alone.

In table 6.0.1 we show the results of all our suggested similarity measures.

embedding	distance	coarsest score	coarse score	fine score
Tfidf-weighted bow	cosine	0.6807	0.5985	0.7360
Text topics	cosine	0.7930	0.7209	0.7992
Text topics	euclid	0.7191	0.6888	0.7810
Image features	cosine	0.9430	0.7576	0.8280
Image features	euclid	0.8666	0.7007	0.8016
Conc. text/image feat.	cosine	0.9608	0.8076	**0.8626**
Conc. text/image feat.	euclid	0.9608	0.8076	**0.8626**
Semantic embedding	cosine	**0.9801**	**0.8230**	0.6704
Semantic embedding	euclid	0.9705	0.8219	0.6710

Table 6.0.1: All similarity measures scored on our evaluation sets as described in chapter 2.3.
The best scores for each set are highlighted in bold.

6.1 Applications at *FINN.NO*

There are some implementation issues that may need to be discussed when applying these similarity measures at *finn.no*. Extracting the features from a given ad is quite computationally demanding and usually takes a few seconds, and it is thus not practical to calculate these every time we need them. We instead suggest to extract these features on creation or modification of the classified ads, and store the features in a suitable database. The features can then be retrieved on demand when doing e.g. similarity comparisons.

Calculating similarities for a large set of ads in order to find the most similar ad, may also be computationally costly and take too long time for e.g. real-time web search. It is therefore suggested to use a method that will first find a smaller *candidate set*, and then calculate similarities for the ads in this set. Finding such a candidate set can be done via either locality-sensitive hashing (Charikar, 2002) or learned binary hash codes (Grauman and Fergus, 2013).

The main applications of the similarity measures we've described is in recommendation, i.e. suggesting similar ads that may be of interest to the user,

based on ads the user has browsed recently. However, there may also be applications in error-correction of ads, e.g. correcting erroneous categories the users have assigned to an ad.

Another application is to model the prices of classes of items based on historical prices for items in *finn.no*. In some cases, users putting items for sale on *finn.no* might not have a good idea of what an item should cost. Using our similarity measure, we might suggest to this user what "similar" objects have cost in the past, and thus help the user identify a reasonable price that might either help the object be sold quickly or for as high price as is feasible.

An application of our similarity measure for *images* might be to search *finn.no* by images. As an example *finn.no* has a vast amount of used clothing for sale at any time, but finding the piece of clothing we're interested in may be tedious to do by text alone. If we have a photo of a similar piece of clothing we're interested in, we can search using this photo as an example, and find the clothing which most resembles this photo. A similar approach may be suitable for e.g. furniture. We do however suggest that in order for this to work properly, we may need to train our image classifier on a larger, and more fine-grained, set of image examples of the class we're interested in.

6.2 Improvements and further work

There are numerous ways we might improve on our models. In order for the semantic embedding similarity measure to be more useful for fine-detailed similarity measurement, it would need many more categories than our current set of categories. Separating the ads into a larger number of categories is however a very challenging task, and thus may be too cumbersome to pursue.

For our simple concatenated similarity measure, we might try to increase the information encoded in the embedding via increasing the length of the embedding. For the topic model this means increasing the number of inferred topics, and for the image part, we might investigate architectures with more features in the final layer. There also exist methods which allows us to learn a bilinear similarity measure (e.g. Chechik et al. (2010)) based on our image and text features, which might be a fruitful route of investigation. A more principled way of learning similarity measures for images and text might be to learn the embedding via optimizing a

suitable loss function, such as in the *deep ranking* model of Wang et al. (2014).

There are also many other ways of modeling the text which might be suitable for an embedding, such as *paragraph vectors* (Le and Mikolov, 2014) or *recurrent neural network* encoders (Vinyals et al., 2014). This type of modeling has spawned a lot of interest recently in the field of *image captioning*, see e.g. Vinyals et al. (2014), Karpathy and Fei-Fei (2014) and Kiros et al. (2014). Some of these methods might be applicable to multi-modal settings such as ours with some modification.

We've focused on learning a similarity measure which can be useful in a number of tasks such as e.g. clustering items. However, the most common use for our similarity measure will be in ranking items according to similarity and providing the top n most similar items for e.g. recommendations to a user. A drawback with our scoring in this setting, is that it focuses too much on how well the similarity measure manages to rank *all* items in a list, rather than the quality or "relevance" of the *top* ranked items, which in these settings is the most important part. It might not seem obvious that this is a drawback, but e.g. the tf-idf weighted bag-of-words embedding, which we describe in chapter 4.1, tends to perform very well at finding the top most similar items, but poorly at ranking correctly items further down the list, which severely impacts its score in our evaluation method. Thus our evaluation method does not truly reflect the quality of the similarity measures on ranking tasks where we're mostly just interested in the top results.

As a small test, we compared the results from our concatenated embedding in (5.1) with another version where we replaced the topic embedding with tf-idf bag-of-word embeddings. In almost all cases, the embedding based on tf-idf features gave more relevant top results than the embedding based on topic features. We thus suggest that to properly evaluate our image-text embeddings for such ranking tasks, we need to score our similarity measures by other metrics, such as e.g. *top-n precision* or *top-n discounted cumulative gain* (see Hang (2011) for definition of these metrics). These metrics do however require a labeling of what can be considered the most "relevant" items according to specific similarity comparisons. We thus suggest that a first step in further evaluation of our similarity measures for top-n ranking purposes would be to create such labellings, either through manual annotation or other means.

7 Appendix

7.1 Finn-categories and selections

category	selected	class number
Antikviteter og kunst/Andre antikviteter		
Antikviteter og kunst/Antikke møbler		
Antikviteter og kunst/Keramikk, porselen og glass		
Antikviteter og kunst/Kunst		
Antikviteter og kunst/Sølvtøy og bestikk	×	151
Dyr og utstyr/Akvarier		
Dyr og utstyr/Andre dyr		
Dyr og utstyr/Annet dyreutstyr		
Dyr og utstyr/Bur	×	3
Dyr og utstyr/Fisker	×	20
Dyr og utstyr/Fôrverter, dyrepass, avl og stallplasser		
Dyr og utstyr/Fugler	×	9
Dyr og utstyr/Heste- og rideutstyr		
Dyr og utstyr/Hester/Kaldblodshester	×	0
Dyr og utstyr/Hester/Ponnier	×	2
Dyr og utstyr/Hester/Varmblodshester	×	1
Dyr og utstyr/Hunder/Apporterende hunder	×	18
Dyr og utstyr/Hunder/Blandingshunder	×	10
Dyr og utstyr/Hunder/Bruks-, hyrde- og gjeterhunder	×	14
Dyr og utstyr/Hunder/Dachs-, drivende og sporhunder	×	13

Dyr og utstyr/Hunder/Mynder	×	12
Dyr og utstyr/Hunder/Pinscher-, schnauzer-, molosser og sennenhunder	×	15
Dyr og utstyr/Hunder/Selskapshunder	×	16
Dyr og utstyr/Hunder/Spisshunder	×	11
Dyr og utstyr/Hunder/Stående fuglehunder	×	19
Dyr og utstyr/Hunder/Terriere	×	17
Dyr og utstyr/Hundeutstyr		
Dyr og utstyr/Katter/Blandingskatter	×	7
Dyr og utstyr/Katter/Korthåret	×	5
Dyr og utstyr/Katter/Orientalere	×	4
Dyr og utstyr/Katter/Persere	×	8
Dyr og utstyr/Katter/Semi-langhåret	×	6
Dyr og utstyr/Katteutstyr		
Elektronikk og hvitevarer/Data/Bærbar PC	×	45
Elektronikk og hvitevarer/Data/Datatilbehør/Harddisk/lagring		
Elektronikk og hvitevarer/Data/Datatilbehør/Komponenter		
Elektronikk og hvitevarer/Data/Datatilbehør/Printere og tilbehør		
Elektronikk og hvitevarer/Data/Datatilbehør/Tilleggsutstyr		
Elektronikk og hvitevarer/Data/Datatilbehør/Annet		
Elektronikk og hvitevarer/Data/Kalkulatorer	×	47
Elektronikk og hvitevarer/Data/Programvare		
Elektronikk og hvitevarer/Data/Skjermer	×	46
Elektronikk og hvitevarer/Data/Stasjonær PC	×	43
Elektronikk og hvitevarer/Data/Tablet og lesebrett	×	44
Elektronikk og hvitevarer/Foto og video/Annet fotoutstyr		
Elektronikk og hvitevarer/Foto og video/Fotovesker og -bager	×	53
Elektronikk og hvitevarer/Foto og video/Hybridkamera	×	50
Elektronikk og hvitevarer/Foto og video/Kompaktkameraer	×	49
Elektronikk og hvitevarer/Foto og video/Objektiver	×	51
Elektronikk og hvitevarer/Foto og video/Systemkameraer	×	48
Elektronikk og hvitevarer/Foto og video/Videokameraer	×	52
Elektronikk og hvitevarer/Husholdningsapparater/Andre apparater		
Elektronikk og hvitevarer/Husholdningsapparater/Blender	×	34
Elektronikk og hvitevarer/Husholdningsapparater/Brødrister	×	40
Elektronikk og hvitevarer/Husholdningsapparater/Kaffemaskin	×	42
Elektronikk og hvitevarer/Husholdningsapparater/Kjøkkenmaskin og foodprosessor	×	38
Elektronikk og hvitevarer/Husholdningsapparater/Miksere	×	37
Elektronikk og hvitevarer/Husholdningsapparater/Strykejern	×	36
Elektronikk og hvitevarer/Husholdningsapparater/Støvsuger	×	41
Elektronikk og hvitevarer/Husholdningsapparater/Vaffel- og toastjern	×	39
Elektronikk og hvitevarer/Husholdningsapparater/Vannkoker	×	35
Elektronikk og hvitevarer/Hvitevarer/Andre hvitevarer		
Elektronikk og hvitevarer/Hvitevarer/Frysere/Annet		
Elektronikk og hvitevarer/Hvitevarer/Frysere/Fryseboks	×	22
Elektronikk og hvitevarer/Hvitevarer/Frysere/Fryseskap	×	21
Elektronikk og hvitevarer/Hvitevarer/Innbyggingsovner	×	30
Elektronikk og hvitevarer/Hvitevarer/Kjøleskap	×	28
Elektronikk og hvitevarer/Hvitevarer/Komfyrer	×	26
Elektronikk og hvitevarer/Hvitevarer/Mikrobølgeovner	×	27
Elektronikk og hvitevarer/Hvitevarer/Oppvaskmaskiner	×	24
Elektronikk og hvitevarer/Hvitevarer/Platetopper	×	29
Elektronikk og hvitevarer/Hvitevarer/Tørketromler	×	31

Elektronikk og hvitevarer/Hvitevarer/Vaskemaskiner	×	25
Elektronikk og hvitevarer/Hvitevarer/Ventilatorer	×	23
Elektronikk og hvitevarer/Lyd og bilde/Bluray-spillere	×	57
Elektronikk og hvitevarer/Lyd og bilde/Digital-TV og mediabokser	×	59
Elektronikk og hvitevarer/Lyd og bilde/DVD-spillere	×	67
Elektronikk og hvitevarer/Lyd og bilde/Forsterkere og receivere	×	62
Elektronikk og hvitevarer/Lyd og bilde/Hjemmekinoanlegg	×	60
Elektronikk og hvitevarer/Lyd og bilde/Hodetelefoner	×	63
Elektronikk og hvitevarer/Lyd og bilde/Høyttalere	×	55
Elektronikk og hvitevarer/Lyd og bilde/Kabler og tilbehør		
Elektronikk og hvitevarer/Lyd og bilde/MP3 og bærbar lyd	×	56
Elektronikk og hvitevarer/Lyd og bilde/PA-utstyr		
Elektronikk og hvitevarer/Lyd og bilde/Projektor og lerret	×	61
Elektronikk og hvitevarer/Lyd og bilde/Radio	×	58
Elektronikk og hvitevarer/Lyd og bilde/Stereo/Annet		
Elektronikk og hvitevarer/Lyd og bilde/Stereo/CD-spillere	×	66
Elektronikk og hvitevarer/Lyd og bilde/Stereo/Platespillere	×	65
Elektronikk og hvitevarer/Lyd og bilde/Stereo/Radio		
Elektronikk og hvitevarer/Lyd og bilde/Stereo/Stereopakker		
Elektronikk og hvitevarer/Lyd og bilde/TV	×	64
Elektronikk og hvitevarer/Lyd og bilde/Videospillere	×	54
Elektronikk og hvitevarer/Spill og konsoll/Spill	×	69
Elektronikk og hvitevarer/Spill og konsoll/Spillkonsoller	×	68
Elektronikk og hvitevarer/Spill og konsoll/Tilbehør		
Elektronikk og hvitevarer/Telefoner og tilbehør/Andre telefoner	×	32
Elektronikk og hvitevarer/Telefoner og tilbehør/Mobiltelefoner	×	33
Elektronikk og hvitevarer/Telefoner og tilbehør/Telefontilbehør		
Elektronikk og hvitevarer/Annet		
Foreldre og barn/Barneklær og sko/Baby	×	264
Foreldre og barn/Barneklær og sko/Gutt	×	262
Foreldre og barn/Barneklær og sko/Jente	×	263
Foreldre og barn/Barnemøbler/Annet		
Foreldre og barn/Barnemøbler/Bord		
Foreldre og barn/Barnemøbler/Oppbevaring		
Foreldre og barn/Barnemøbler/Senger	×	269
Foreldre og barn/Barnemøbler/Stellebord	×	268
Foreldre og barn/Barnemøbler/Stoler	×	267
Foreldre og barn/Barneseter	×	265
Foreldre og barn/Barnevogner	×	266
Foreldre og barn/Leker/Andre leker		
Foreldre og barn/Leker/Babyleker		
Foreldre og barn/Leker/Bamser, dukker og figurer		
Foreldre og barn/Leker/Biler og baner		
Foreldre og barn/Leker/Lego og byggeklosser		
Foreldre og barn/Leker/Tegne- og formingsutstyr		
Foreldre og barn/Leker/Uteleker		
Foreldre og barn/Mammaklær		
Foreldre og barn/Utstyr og sikkerhet		
Foreldre og barn/Annet		
Fritid, hobby og underholdning/Billetter og reiser/Flybilletter		
Fritid, hobby og underholdning/Billetter og reiser/Idrettsarrangementer		
Fritid, hobby og underholdning/Billetter og reiser/Konsertbilletter		

Fritid, hobby og underholdning/Billetter og reiser/Pakkereiser		
Fritid, hobby og underholdning/Billetter og reiser/Teaterbilletter		
Fritid, hobby og underholdning/Billetter og reiser/Togbilletter		
Fritid, hobby og underholdning/Billetter og reiser/Annet		
Fritid, hobby og underholdning/Brettspill og bordspill/		
Fritid, hobby og underholdning/Brettspill og bordspill/Andre spill		
Fritid, hobby og underholdning/Brettspill og bordspill/Bordspill		
Fritid, hobby og underholdning/Brettspill og bordspill/Brettspill		
Fritid, hobby og underholdning/Brettspill og bordspill/Hage- og parkspill		
Fritid, hobby og underholdning/Brettspill og bordspill/Puslespill		
Fritid, hobby og underholdning/Bøker og blader/Blader og magasiner	×	94
Fritid, hobby og underholdning/Bøker og blader/Faglitteratur	×	97
Fritid, hobby og underholdning/Bøker og blader/Fakta og dokumentar	×	95
Fritid, hobby og underholdning/Bøker og blader/Kokebøker	×	96
Fritid, hobby og underholdning/Bøker og blader/Oppslagsverk	×	93
Fritid, hobby og underholdning/Bøker og blader/Skjønnlitteratur	×	92
Fritid, hobby og underholdning/Bøker og blader/Øvrige bøker og verk	×	91
Fritid, hobby og underholdning/Bøker og blader/Tegneserier	×	98
Fritid, hobby og underholdning/Håndarbeid/Garn	×	112
Fritid, hobby og underholdning/Håndarbeid/Keramikk	×	109
Fritid, hobby og underholdning/Håndarbeid/Maling og maleutstyr		
Fritid, hobby og underholdning/Håndarbeid/Perler og smykkestener		
Fritid, hobby og underholdning/Håndarbeid/Scrapbooking		
Fritid, hobby og underholdning/Håndarbeid/Stoff		
Fritid, hobby og underholdning/Håndarbeid/Sy- og strikkeutstyr		
Fritid, hobby og underholdning/Håndarbeid/Symaskiner	×	110
Fritid, hobby og underholdning/Håndarbeid/Tresløyd		
Fritid, hobby og underholdning/Håndarbeid/Vev	×	111
Fritid, hobby og underholdning/Mat og drikke		
Fritid, hobby og underholdning/Modellbyggesett og modeller		
Fritid, hobby og underholdning/Musikk og film/Blu-ray	×	101
Fritid, hobby og underholdning/Musikk og film/CD	×	99
Fritid, hobby og underholdning/Musikk og film/DVD	×	100
Fritid, hobby og underholdning/Musikk og film/LP/EP	×	103
Fritid, hobby og underholdning/Musikk og film/VHS	×	102
Fritid, hobby og underholdning/Musikkinstrumenter/Andre strengeinstrumenter	×	80
Fritid, hobby og underholdning/Musikkinstrumenter/Bassgitarer	×	85
Fritid, hobby og underholdning/Musikkinstrumenter/Gitarer	×	83
Fritid, hobby og underholdning/Musikkinstrumenter/Keyboard/synth	×	81
Fritid, hobby og underholdning/Musikkinstrumenter/Lydutstyr		
Fritid, hobby og underholdning/Musikkinstrumenter/Messinginstrumenter	×	88
Fritid, hobby og underholdning/Musikkinstrumenter/Orgler	×	87
Fritid, hobby og underholdning/Musikkinstrumenter/Piano/flygel	×	86
Fritid, hobby og underholdning/Musikkinstrumenter/Slagverk	×	90
Fritid, hobby og underholdning/Musikkinstrumenter/Strykeinstrumenter	×	84
Fritid, hobby og underholdning/Musikkinstrumenter/Treblåsere	×	89
Fritid, hobby og underholdning/Musikkinstrumenter/Trekkspill	×	82
Fritid, hobby og underholdning/Musikkinstrumenter/Øvrige instrumenter		
Fritid, hobby og underholdning/RC-utstyr/Bil	×	108
Fritid, hobby og underholdning/RC-utstyr/Båt	×	106
Fritid, hobby og underholdning/RC-utstyr/Fly	×	107
Fritid, hobby og underholdning/RC-utstyr/Helikopter		

Fritid, hobby og underholdning/RC-utstyr/Modelljernbaner		
Fritid, hobby og underholdning/Samleobjekter/Andre samlinger		
Fritid, hobby og underholdning/Samleobjekter/Frimerker	×	105
Fritid, hobby og underholdning/Samleobjekter/Mynter og sedler	×	104
Fritid, hobby og underholdning/Samleobjekter/Pins		
Fritid, hobby og underholdning/Samleobjekter/Postkort		
Fritid, hobby og underholdning/Annet		
Hage, oppussing og hus/Alarm og sikkerhet		
Hage, oppussing og hus/Baderomsinnredning/Badekar	×	185
Hage, oppussing og hus/Baderomsinnredning/Baderomsmøbler	×	186
Hage, oppussing og hus/Baderomsinnredning/Badstuer		
Hage, oppussing og hus/Baderomsinnredning/Blandebatterier	×	184
Hage, oppussing og hus/Baderomsinnredning/Dusjkabinett og -vegger	×	180
Hage, oppussing og hus/Baderomsinnredning/Servanter	×	181
Hage, oppussing og hus/Baderomsinnredning/Steamdusjer	×	183
Hage, oppussing og hus/Baderomsinnredning/Tilbehør		
Hage, oppussing og hus/Baderomsinnredning/Toaletter	×	182
Hage, oppussing og hus/Byggevarer og oppussing/Andre byggevarer		
Hage, oppussing og hus/Byggevarer og oppussing/Byggesett	×	163
Hage, oppussing og hus/Byggevarer og oppussing/Dører	×	168
Hage, oppussing og hus/Byggevarer og oppussing/Gulv	×	167
Hage, oppussing og hus/Byggevarer og oppussing/Tapet og maling	×	164
Hage, oppussing og hus/Byggevarer og oppussing/Trapper	×	166
Hage, oppussing og hus/Byggevarer og oppussing/Vinduer	×	165
Hage, oppussing og hus/Garasje	×	169
Hage, oppussing og hus/Hage og uteområder/Annet hageutstyr		
Hage, oppussing og hus/Hage og uteområder/Beplantning og tilbehør		
Hage, oppussing og hus/Hage og uteområder/Gressklippere	×	175
Hage, oppussing og hus/Hage og uteområder/Grill	×	172
Hage, oppussing og hus/Hage og uteområder/Hageredskap	×	173
Hage, oppussing og hus/Hage og uteområder/Hagemøbler		
Hage, oppussing og hus/Hage og uteområder/Markiser og parasoller		
Hage, oppussing og hus/Hage og uteområder/Partytelt	×	174
Hage, oppussing og hus/Hage og uteområder/Snørydding	×	171
Hage, oppussing og hus/Hage og uteområder/Svømmebasseng og spabad		
Hage, oppussing og hus/Hytteutstyr		
Hage, oppussing og hus/Kjøkkeninnredning	×	170
Hage, oppussing og hus/Oppvarming og ventilasjon/Aircondition og vifter		
Hage, oppussing og hus/Oppvarming og ventilasjon/Elektriske ovner	×	177
Hage, oppussing og hus/Oppvarming og ventilasjon/Ovner	×	179
Hage, oppussing og hus/Oppvarming og ventilasjon/Peis	×	176
Hage, oppussing og hus/Oppvarming og ventilasjon/Varmepumper		
Hage, oppussing og hus/Oppvarming og ventilasjon/Ved og brensel	×	178
Hage, oppussing og hus/Verktøy		
Hage, oppussing og hus/Annet		
Klær, kosmetikk og accessoirer/Briller og linser/Briller	×	260
Klær, kosmetikk og accessoirer/Briller og linser/Linser		
Klær, kosmetikk og accessoirer/Briller og linser/Solbriller	×	259
Klær, kosmetikk og accessoirer/Dameklær/Annet		
Klær, kosmetikk og accessoirer/Dameklær/Brudekjoler	×	249
Klær, kosmetikk og accessoirer/Dameklær/Bukser	×	246
Klær, kosmetikk og accessoirer/Dameklær/Bunader	×	257

Klær, kosmetikk og accessoirer/Dameklær/Gensere	×	253
Klær, kosmetikk og accessoirer/Dameklær/Jakker	×	247
Klær, kosmetikk og accessoirer/Dameklær/Kjoler	×	251
Klær, kosmetikk og accessoirer/Dameklær/Luer,skjerf og votter	×	248
Klær, kosmetikk og accessoirer/Dameklær/Skjorter	×	255
Klær, kosmetikk og accessoirer/Dameklær/Skjørt	×	256
Klær, kosmetikk og accessoirer/Dameklær/T-skjorter	×	245
Klær, kosmetikk og accessoirer/Dameklær/Topper	×	252
Klær, kosmetikk og accessoirer/Dameklær/Undertøy	×	250
Klær, kosmetikk og accessoirer/Dameklær/Ytterklær	×	254
Klær, kosmetikk og accessoirer/Herreklær/Annet		
Klær, kosmetikk og accessoirer/Herreklær/Bukser	×	234
Klær, kosmetikk og accessoirer/Herreklær/Bunader	×	242
Klær, kosmetikk og accessoirer/Herreklær/Dresser	×	240
Klær, kosmetikk og accessoirer/Herreklær/Gensere	×	238
Klær, kosmetikk og accessoirer/Herreklær/Luer,skjerf og votter	×	235
Klær, kosmetikk og accessoirer/Herreklær/Skjorter	×	241
Klær, kosmetikk og accessoirer/Herreklær/Smokinger	×	237
Klær, kosmetikk og accessoirer/Herreklær/T-skjorter	×	233
Klær, kosmetikk og accessoirer/Herreklær/Undertøy	×	236
Klær, kosmetikk og accessoirer/Herreklær/Ytterklær	×	239
Klær, kosmetikk og accessoirer/Hud-, hår- og kroppspleie		
Klær, kosmetikk og accessoirer/Klokker og ur	×	258
Klær, kosmetikk og accessoirer/Kostyme		
Klær, kosmetikk og accessoirer/Kosmetikk	×	261
Klær, kosmetikk og accessoirer/Sko/Damesko	×	243
Klær, kosmetikk og accessoirer/Sko/Herresko	×	244
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Andre smykker		
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Armbånd	×	229
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Halskjeder	×	231
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Ringer	×	230
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Smykkeskrin og oppbevaring	×	232
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Ørepynt	×	228
Klær, kosmetikk og accessoirer/Annet		
Klær, kosmetikk og accessoirer/Vesker og bager	×	227
Møbler og interiør/Annet		
Møbler og interiør/Belysning/Andre lamper		
Møbler og interiør/Belysning/Bordlamper	×	136
Møbler og interiør/Belysning/Stålamper	×	137
Møbler og interiør/Belysning/Taklamper	×	135
Møbler og interiør/Belysning/Vegglamper	×	138
Møbler og interiør/Bord og stoler/Andre bord og stoler		
Møbler og interiør/Bord og stoler/Kjøkkenbord	×	129
Møbler og interiør/Bord og stoler/Kontorstoler	×	130
Møbler og interiør/Bord og stoler/Salongbord	×	125
Møbler og interiør/Bord og stoler/Skrivebord	×	128
Møbler og interiør/Bord og stoler/Spisestuer	×	126
Møbler og interiør/Bord og stoler/Stoler og krakker	×	127
Møbler og interiør/Dekorasjon og pyntegjenstander/Andre pyntgjenstander		
Møbler og interiør/Dekorasjon og pyntegjenstander/Bilder og rammer	×	121
Møbler og interiør/Dekorasjon og pyntegjenstander/Fat	×	124
Møbler og interiør/Dekorasjon og pyntegjenstander/Lysestaker	×	120

Møbler og interiør/Dekorasjon og pyntegjenstander/Speil	×	123
Møbler og interiør/Dekorasjon og pyntegjenstander/Vaser	×	122
Møbler og interiør/Garderobe og skap/Garderober	×	145
Møbler og interiør/Garderobe og skap/Seksjoner	×	146
Møbler og interiør/Garderobe og skap/Skap	×	144
Møbler og interiør/Hyller og kommoder/Hyller	×	139
Møbler og interiør/Hyller og kommoder/Kommoder	×	143
Møbler og interiør/Hyller og kommoder/Nattbord	×	142
Møbler og interiør/Hyller og kommoder/Skjenk	×	140
Møbler og interiør/Hyller og kommoder/TV-møbler	×	141
Møbler og interiør/Kjøkkenutstyr og serviser	×	134
Møbler og interiør/Senger og madrasser/Madrasser	×	131
Møbler og interiør/Senger og madrasser/Rammemadrasser	×	133
Møbler og interiør/Senger og madrasser/Senger	×	132
Møbler og interiør/Sofaer og lenestoler/2-seter	×	113
Møbler og interiør/Sofaer og lenestoler/3-seter	×	118
Møbler og interiør/Sofaer og lenestoler/Hjørnesofaer	×	115
Møbler og interiør/Sofaer og lenestoler/Lenestoler	×	116
Møbler og interiør/Sofaer og lenestoler/Puffer	×	117
Møbler og interiør/Sofaer og lenestoler/Sofagrupper	×	114
Møbler og interiør/Sofaer og lenestoler/Sovesofaer	×	119
Møbler og interiør/Tepper og tekstiler/Andre tekstiler		
Møbler og interiør/Tepper og tekstiler/Duker	×	147
Møbler og interiør/Tepper og tekstiler/Gardiner	×	148
Møbler og interiør/Tepper og tekstiler/Puter	×	149
Møbler og interiør/Tepper og tekstiler/Tepper	×	150
Næringsvirksomhet/Butikk og varehandel/Annet butikkutstyr		
Næringsvirksomhet/Butikk og varehandel/Betalingsutstyr	×	73
Næringsvirksomhet/Butikk og varehandel/Butikkbelysning		
Næringsvirksomhet/Butikk og varehandel/Butikkinnredning		
Næringsvirksomhet/Butikk og varehandel/Lagerutstyr		
Næringsvirksomhet/Butikk og varehandel/Sikkerhet og overvåkning	×	72
Næringsvirksomhet/Butikk og varehandel/Vareparti og konkursbo		
Næringsvirksomhet/Container og brakker/Annet anleggsutstyr		
Næringsvirksomhet/Container og brakker/Brakker	×	71
Næringsvirksomhet/Container og brakker/Containere	×	70
Næringsvirksomhet/Container og brakker/Mannskapsvogner		
Næringsvirksomhet/Helse og behandling/Annet utstyr		
Næringsvirksomhet/Helse og behandling/Førstehjelpsutstyr		
Næringsvirksomhet/Helse og behandling/Hjelpemidler	×	78
Næringsvirksomhet/Helse og behandling/Hud- og kroppspleie		
Næringsvirksomhet/Helse og behandling/Utstyr og inventar		
Næringsvirksomhet/Kontorutstyr og innredning/Innredning		
Næringsvirksomhet/Kontorutstyr og innredning/Kopimaskiner	×	77
Næringsvirksomhet/Kontorutstyr og innredning/Printere	×	76
Næringsvirksomhet/Kontorutstyr og innredning/Rekvisita		
Næringsvirksomhet/Landbruk		
Næringsvirksomhet/Last og transport		
Næringsvirksomhet/Maskinutstyr og reservedeler		
Næringsvirksomhet/Scene	×	79
Næringsvirksomhet/Storkjøkken og restaurant/Annet utstyr		
Næringsvirksomhet/Storkjøkken og restaurant/Innredning		

Næringsvirksomhet/Storkjøkken og restaurant/Kjøkkenutstyr		
Næringsvirksomhet/Storkjøkken og restaurant/Kjøl og frys		
Næringsvirksomhet/Storkjøkken og restaurant/Koke- og stekeprodukter		
Næringsvirksomhet/Storkjøkken og restaurant/Oppvask		
Næringsvirksomhet/Verksted, bygg og anlegg/Annet utstyr		
Næringsvirksomhet/Verksted, bygg og anlegg/Byggtørkere	×	74
Næringsvirksomhet/Verksted, bygg og anlegg/Kompressor		
Næringsvirksomhet/Verksted, bygg og anlegg/Lagerhall		
Næringsvirksomhet/Verksted, bygg og anlegg/Lys		
Næringsvirksomhet/Verksted, bygg og anlegg/Maskiner og utstyr		
Næringsvirksomhet/Verksted, bygg og anlegg/Maskinutleie		
Næringsvirksomhet/Verksted, bygg og anlegg/Stillas og reoler	×	73
Næringsvirksomhet/Verksted, bygg og anlegg/Treforedling		
Næringsvirksomhet/Verksted, bygg og anlegg/Verkstedsutstyr		
Næringsvirksomhet/Verksted, bygg og anlegg/Verktøy		
Næringsvirksomhet/Webdomener og gullnummer		
Næringsvirksomhet/Annet		
Sport og friluftsliv/Annen sport		
Sport og friluftsliv/Ballsport/Annen ballsport		
Sport og friluftsliv/Ballsport/Bandy	×	194
Sport og friluftsliv/Ballsport/Basketball	×	193
Sport og friluftsliv/Ballsport/Fotball	×	196
Sport og friluftsliv/Ballsport/Håndball		
Sport og friluftsliv/Ballsport/Tennis	×	195
Sport og friluftsliv/Ballsport/Volleyball		
Sport og friluftsliv/Ekstremsport/Fallskjermhopping		
Sport og friluftsliv/Ekstremsport/Kiting		
Sport og friluftsliv/Ekstremsport/Klatring		
Sport og friluftsliv/Ekstremsport/Skate- og longboard	×	217
Sport og friluftsliv/Golf	×	216
Sport og friluftsliv/GPS og pulsklokker	×	218
Sport og friluftsliv/Jakt, fiske og friluftsliv		
Sport og friluftsliv/Jakt, fiske og friluftsliv/Annet fiskeutstyr		
Sport og friluftsliv/Jakt, fiske og friluftsliv/Annet jaktutstyr		
Sport og friluftsliv/Jakt, fiske og friluftsliv/Annet turutstyr		
Sport og friluftsliv/Jakt, fiske og friluftsliv/Fiskeklær		
Sport og friluftsliv/Jakt, fiske og friluftsliv/Fiskestenger	×	220
Sport og friluftsliv/Jakt, fiske og friluftsliv/Jaktklær		
Sport og friluftsliv/Jakt, fiske og friluftsliv/Kano	×	224
Sport og friluftsliv/Jakt, fiske og friluftsliv/Kikkerter og optikk	×	222
Sport og friluftsliv/Jakt, fiske og friluftsliv/Ryggsekker	×	223
Sport og friluftsliv/Jakt, fiske og friluftsliv/Sneller, fluer og kroker	×	219
Sport og friluftsliv/Jakt, fiske og friluftsliv/Soveposer	×	226
Sport og friluftsliv/Jakt, fiske og friluftsliv/Teiner og garn		
Sport og friluftsliv/Jakt, fiske og friluftsliv/Telt	×	225
Sport og friluftsliv/Jakt, fiske og friluftsliv/Våpen	×	221
Sport og friluftsliv/Kosttilskudd		
Sport og friluftsliv/Skisport/Alpint	×	214
Sport og friluftsliv/Skisport/Annet skiutstyr		
Sport og friluftsliv/Skisport/Langrenn	×	215
Sport og friluftsliv/Skisport/Snowboard	×	213
Sport og friluftsliv/Skisport/Telemark	×	212

Sport og friluftsliv/Skøytesport	×	204
Sport og friluftsliv/Sportsklær og sko		
Sport og friluftsliv/Sportsskyting/Luftvåpen	×	199
Sport og friluftsliv/Sportsskyting/Paintball	×	198
Sport og friluftsliv/Sportsskyting/Pistol		
Sport og friluftsliv/Sportsskyting/Softgun	×	197
Sport og friluftsliv/Startnummer		
Sport og friluftsliv/Supporterutstyr		
Sport og friluftsliv/Sykkelsport/Sykler/Annet		
Sport og friluftsliv/Sykkelsport/Sykler/BMX	×	207
Sport og friluftsliv/Sykkelsport/Sykler/Cyclocross	×	206
Sport og friluftsliv/Sykkelsport/Sykler/Elektriske	×	209
Sport og friluftsliv/Sykkelsport/Sykler/Fulldemper	×	205
Sport og friluftsliv/Sykkelsport/Sykler/Hybrid/bysykkel	×	208
Sport og friluftsliv/Sykkelsport/Sykler/Landevei	×	210
Sport og friluftsliv/Sykkelsport/Sykler/Terreng	×	211
Sport og friluftsliv/Sykkelsport/Utstyr		
Sport og friluftsliv/Treningsapparater og -utstyr/Ellipsemaskiner	×	189
Sport og friluftsliv/Treningsapparater og -utstyr/Romaskiner	×	188
Sport og friluftsliv/Treningsapparater og -utstyr/Stepmaskiner	×	192
Sport og friluftsliv/Treningsapparater og -utstyr/Styrkeapparater	×	190
Sport og friluftsliv/Treningsapparater og -utstyr/Tredemøller	×	187
Sport og friluftsliv/Treningsapparater og -utstyr/Treningssykler	×	191
Sport og friluftsliv/Treningsapparater og -utstyr/Treningsutstyr		
Sport og friluftsliv/Vannsport/Annet utstyr		
Sport og friluftsliv/Vannsport/Dykking	×	200
Sport og friluftsliv/Vannsport/Padling	×	201
Sport og friluftsliv/Vannsport/Seiling		
Sport og friluftsliv/Vannsport/Surfing	×	202
Sport og friluftsliv/Vannsport/Windsurf	×	203
Utstyr til bil, båt og MC/ATV-deler		
Utstyr til bil, båt og MC/Bildeler/Annet biltilbehør		
Utstyr til bil, båt og MC/Bildeler/Bildeler		
Utstyr til bil, båt og MC/Bildeler/Bilseter	×	152
Utstyr til bil, båt og MC/Bildeler/Bilstereo		
Utstyr til bil, båt og MC/Bildeler/Dekk og felger	×	155
Utstyr til bil, båt og MC/Bildeler/GPS	×	154
Utstyr til bil, båt og MC/Bildeler/Styling		
Utstyr til bil, båt og MC/Bildeler/Takstativ og boks	×	153
Utstyr til bil, båt og MC/Båtdeler/Annet båtutstyr		
Utstyr til bil, båt og MC/Båtdeler/Elektronikk		
Utstyr til bil, båt og MC/Båtdeler/Fortøyningsutstyr		
Utstyr til bil, båt og MC/Båtdeler/Interiør og eksteriør		
Utstyr til bil, båt og MC/Båtdeler/Motordeler		
Utstyr til bil, båt og MC/Båtdeler/Navigasjon		
Utstyr til bil, båt og MC/Båtdeler/Seilbåtutstyr		
Utstyr til bil, båt og MC/Båtdeler/Sikkerhet		
Utstyr til bil, båt og MC/Båtdeler/Strøm og VVS		
Utstyr til bil, båt og MC/Båtdeler/Styring og vinsj		
Utstyr til bil, båt og MC/Caravandeler		
Utstyr til bil, båt og MC/MC-deler/MC-deler		
Utstyr til bil, båt og MC/MC-deler/MC-klær	×	162

Utstyr til bil, båt og MC/MC-deler/MC-tilbehør		
Utstyr til bil, båt og MC/Tilhengere/ATV- og MC-hengere		
Utstyr til bil, båt og MC/Tilhengere/Biltransporthengere	×	158
Utstyr til bil, båt og MC/Tilhengere/Båt- og opplagshengere	×	157
Utstyr til bil, båt og MC/Tilhengere/Hestehengere	×	156
Utstyr til bil, båt og MC/Tilhengere/Maskinhengere		
Utstyr til bil, båt og MC/Tilhengere/Skaphengere	×	160
Utstyr til bil, båt og MC/Tilhengere/Tilhengertilbehør		
Utstyr til bil, båt og MC/Tilhengere/Tipphengere	×	159
Utstyr til bil, båt og MC/Tilhengere/Varehengere	×	161
Utstyr til bil, båt og MC/Annet		

7.2 Mappings between categories and word-embeddings

category	word-embeddings
Dyr og utstyr/Hester/Kaldblodshester	kaldblodshest, kaldblodshester, kaldblods, fjording, fjordhest
Dyr og utstyr/Hester/Varmblodshester	varmblods, varmblods ridehest, varmblodstraver, travhest
Dyr og utstyr/Hester/Ponnier	ponni, ponnier
Dyr og utstyr/Bur	bur,fuglebur,transportbur
Dyr og utstyr/Katter/Orientalere	siameser
Dyr og utstyr/Katter/Korthåret	korthåret, britisk korthår, bengal, devon rex, sphynx
Dyr og utstyr/Katter/Semi-langhåret	hellig birma,ragdoll,maine coon
Dyr og utstyr/Katter/Blandingskatter	skogkatt,huskatt
Dyr og utstyr/Katter/Persere	perser
Dyr og utstyr/Fugler	fugl,fugler,undulat,papegøye
Dyr og utstyr/Hunder/Blandingshunder	blandings,blandingshund
Dyr og utstyr/Hunder/Spisshunder	husky,elghund,samojed,buhund,collie
Dyr og utstyr/Hunder/Mynder	whippet,italiensk mynde
Dyr og utstyr/Hunder/Dachs-, drivende og sporhunder	dachs
Dyr og utstyr/Hunder/Bruks-, hyrde- og gjeterhunder	border collie,schæferhund, schæfer,shetland sheepdog,gjeterhund
Dyr og utstyr/Hunder/Pinscher-, schnauzer-, molosser og sennenhunder	rottweiler,mastiff,dobermann, berner sennen,bulldog
Dyr og utstyr/Hunder/Selskapshunder	puddel,chihuahua,tibetansk spaniel, bichon frise,fransk bulldog, dvergpuddel,cavalier king
Dyr og utstyr/Hunder/Terriere	terrier,russell terrier,jack russel, cairn terrier,bull terrier
Dyr og utstyr/Hunder/Apporterende hunder	labrador,golden retriever,cocker spaniel
Dyr og utstyr/Hunder/Stående fuglehunder	fuglehund,pointer
Dyr og utstyr/Fisker	gullfisk,kampfisk,akvariefisk,fisk
Elektronikk og hvitevarer/Hvitevarer/Frysere/Fryseskap	fryseskap
Elektronikk og hvitevarer/Hvitevarer/Frysere/Fryseboks	fryseboks
Elektronikk og hvitevarer/Hvitevarer/Ventilatorer	ventilator
Elektronikk og hvitevarer/Hvitevarer/Oppvaskmaskiner	oppvaskmaskin
Elektronikk og hvitevarer/Hvitevarer/Vaskemaskiner	vaskemaskin
Elektronikk og hvitevarer/Hvitevarer/Komfyrer	komfyr

Elektronikk og hvitevarer/Hvitevarer/Mikrobølgeovner	mikrobølgeovn,microbølgeovn
Elektronikk og hvitevarer/Hvitevarer/Kjøleskap	kjøleskap
Elektronikk og hvitevarer/Hvitevarer/Platetopper	platetopp
Elektronikk og hvitevarer/Hvitevarer/Innbyggingsovner	innbygningsovn,innbyggingsovn
Elektronikk og hvitevarer/Hvitevarer/Tørketromler	tørketrommel
Elektronikk og hvitevarer/Telefoner og tilbehør/Andre telefoner	fasttelefon,ip-telefon,bredbåndstelefon
Elektronikk og hvitevarer/Telefoner og tilbehør/Mobiltelefoner	mobiltelefon,smarttelefon
Elektronikk og hvitevarer/Husholdningsapparater/Blender	blender
Elektronikk og hvitevarer/Husholdningsapparater/Vannkoker	vannkoker
Elektronikk og hvitevarer/Husholdningsapparater/Strykejern	strykejern
Elektronikk og hvitevarer/Husholdningsapparater/Miksere	hurtigmikser
Elektronikk og hvitevarer/Husholdningsapparater/Kjøkkenmaskin og foodprosessor	foodprosessor,kjøkkenmaskin
Elektronikk og hvitevarer/Husholdningsapparater/Vaffel- og toastjern	vaffeljern,toastjern
Elektronikk og hvitevarer/Husholdningsapparater/Brødrister	brødrister
Elektronikk og hvitevarer/Husholdningsapparater/Støvsuger	støvsuger,håndstøvsuger
Elektronikk og hvitevarer/Husholdningsapparater/Kaffemaskin	kaffemaskin,kaffetrakter,kapselmaskin
Elektronikk og hvitevarer/Data/Stasjonær PC	pc,stasjonær
Elektronikk og hvitevarer/Data/Tablet og lesebrett	tablet,nettbrett
Elektronikk og hvitevarer/Data/Bærbar PC	bærbar,laptop
Elektronikk og hvitevarer/Data/Skjermer	skjerm,dataskjerm,pc-skjerm
Elektronikk og hvitevarer/Data/Kalkulatorer	kalkulator,grafisk kalkulator
Elektronikk og hvitevarer/Foto og video/Systemkameraer	systemkamera,speilreflekskamera
Elektronikk og hvitevarer/Foto og video/Kompaktkameraer	kompaktkamera
Elektronikk og hvitevarer/Foto og video/Hybridkamera	hybridkamera
Elektronikk og hvitevarer/Foto og video/Objektiver	objektiv,linse
Elektronikk og hvitevarer/Foto og video/Videokameraer	videokamera
Elektronikk og hvitevarer/Foto og video/Fotovesker og -bager	fotoveske,kamerabag,kameraveske
Elektronikk og hvitevarer/Lyd og bilde/Videospillere	videospiller,vhs-spiller
Elektronikk og hvitevarer/Lyd og bilde/Høyttalere	høyttaler,høyttalere
Elektronikk og hvitevarer/Lyd og bilde/MP3 og bærbar lyd	mp3-spiller,discman,walkman
Elektronikk og hvitevarer/Lyd og bilde/Bluray-spillere	bluray-spiller
Elektronikk og hvitevarer/Lyd og bilde/Radio	radio
Elektronikk og hvitevarer/Lyd og bilde/Digital-TV og mediabokser	digital-tv,mediasenter
Elektronikk og hvitevarer/Lyd og bilde/Hjemmekinoanlegg	hjemmekino,hjemmekinoanlegg
Elektronikk og hvitevarer/Lyd og bilde/Projektor og lerret	projektor,filmlerret
Elektronikk og hvitevarer/Lyd og bilde/Forsterkere og receivere	forsterker,receiver
Elektronikk og hvitevarer/Lyd og bilde/Hodetelefoner	hodetelefoner,headset,headsett
Elektronikk og hvitevarer/Lyd og bilde/TV	tv
Elektronikk og hvitevarer/Lyd og bilde/Stereo/Platespillere	platespiller
Elektronikk og hvitevarer/Lyd og bilde/Stereo/CD-spillere	cdspiller,cd-spiller
Elektronikk og hvitevarer/Lyd og bilde/DVD-spillere	dvd-spiller
Elektronikk og hvitevarer/Spill og konsoll/Spillkonsoller	spillkonsoll
Elektronikk og hvitevarer/Spill og konsoll/Spill	videospill,konsollspill,dataspill
Næringsvirksomhet/Container og brakker/Containere	container,konteinere
Næringsvirksomhet/Container og brakker/Brakker	brakke,brakker
Næringsvirksomhet/Butikk og varehandel/Sikkerhet og overvåkning	overvåkingskamera
Næringsvirksomhet/Butikk og varehandel/Betalingsutstyr	kassaapparat,kortterminal
Næringsvirksomhet/Verksted, bygg og anlegg/Byggtørkere	byggtørker
Næringsvirksomhet/Verksted, bygg og anlegg/Stillas og reoler	stillas
Næringsvirksomhet/Kontorutstyr og innredning/Printere	printer
Næringsvirksomhet/Kontorutstyr og innredning/Kopimaskiner	kopimaskin
Næringsvirksomhet/Helse og behandling/Hjelpemidler	rullestol,rullator

Næringsvirksomhet/Scene	lyskaster,røykmaskin,miksebord
Fritid, hobby og underholdning/Musikkinstrumenter/Andre strengeinstrumenter	ukulele,banjo,dobro,sitar,mandolin
Fritid, hobby og underholdning/Musikkinstrumenter/Keyboard/synth	synth,synthesizer
Fritid, hobby og underholdning/Musikkinstrumenter/Trekkspill	trekkspill
Fritid, hobby og underholdning/Musikkinstrumenter/Gitarer	gitar,el-gitar
Fritid, hobby og underholdning/Musikkinstrumenter/Strykeinstrumenter	fiolin,cello,bratsj
Fritid, hobby og underholdning/Musikkinstrumenter/Bassgitarer	bassgitar
Fritid, hobby og underholdning/Musikkinstrumenter/Piano/flygel	piano,flygel
Fritid, hobby og underholdning/Musikkinstrumenter/Orgler	orgel
Fritid, hobby og underholdning/Musikkinstrumenter/Messinginstrumenter	trompet,saksofon,trombone,tuba,kornett
Fritid, hobby og underholdning/Musikkinstrumenter/Treblåsere	klarinett,obo
Fritid, hobby og underholdning/Musikkinstrumenter/Slagverk	slagverk,trommesett,trommer,perkusjon
Fritid, hobby og underholdning/Bøker og blader/Øvrige bøker og verk	sangbok,barnebok
Fritid, hobby og underholdning/Bøker og blader/Skjønnlitteratur	skjønnlitteratur
Fritid, hobby og underholdning/Bøker og blader/Oppslagsverk	oppslagsverk,leksikon
Fritid, hobby og underholdning/Bøker og blader/Blader og magasiner	blader
Fritid, hobby og underholdning/Bøker og blader/Fakta og dokumentar	faktabøker
Fritid, hobby og underholdning/Bøker og blader/Kokebøker	kokebøker
Fritid, hobby og underholdning/Bøker og blader/Faglitteratur	fagbøker,faglitteratur
Fritid, hobby og underholdning/Bøker og blader/Tegneserier	tegneserie
Fritid, hobby og underholdning/Musikk og film/CD	cder
Fritid, hobby og underholdning/Musikk og film/DVD	dvder,dvd-plater
Fritid, hobby og underholdning/Musikk og film/Blu-ray	blu-ray
Fritid, hobby og underholdning/Musikk og film/VHS	vhs
Fritid, hobby og underholdning/Musikk og film/LP/EP	lp,vinyl
Fritid, hobby og underholdning/Samleobjekter/Mynter og sedler	mynter,sedler
Fritid, hobby og underholdning/Samleobjekter/Frimerker	frimerke
Fritid, hobby og underholdning/RC-utstyr/Båt	båt,rc
Fritid, hobby og underholdning/RC-utstyr/Fly	quadcopter,helikopter
Fritid, hobby og underholdning/RC-utstyr/Bil	rc-bil,nitrobil
Fritid, hobby og underholdning/Håndarbeid/Keramikk	keramikk
Fritid, hobby og underholdning/Håndarbeid/Symaskiner	symaskin
Fritid, hobby og underholdning/Håndarbeid/Vev	vev
Fritid, hobby og underholdning/Håndarbeid/Garn	garn
Møbler og interiør/Sofaer og lenestoler/2-seter	2-seter
Møbler og interiør/Sofaer og lenestoler/Sofagrupper	sofagruppe
Møbler og interiør/Sofaer og lenestoler/Hjørnesofaer	hjørnesofa
Møbler og interiør/Sofaer og lenestoler/Lenestoler	lenestol
Møbler og interiør/Sofaer og lenestoler/Puffer	puff
Møbler og interiør/Sofaer og lenestoler/3-seter	3-seter
Møbler og interiør/Sofaer og lenestoler/Sovesofaer	sovesofa
Møbler og interiør/Dekorasjon og pyntegjenstander/Lysestaker	lysestake
Møbler og interiør/Dekorasjon og pyntegjenstander/Bilder og rammer	bilde,bilderamme
Møbler og interiør/Dekorasjon og pyntegjenstander/Vaser	vase
Møbler og interiør/Dekorasjon og pyntegjenstander/Speil	speil
Møbler og interiør/Dekorasjon og pyntegjenstander/Fat	fat,serveringsfat
Møbler og interiør/Bord og stoler/Salongbord	salongbord
Møbler og interiør/Bord og stoler/Spisestuer	spisestue,spisegruppe
Møbler og interiør/Bord og stoler/Stoler og krakker	stol,krakk
Møbler og interiør/Bord og stoler/Skrivebord	skrivebord
Møbler og interiør/Bord og stoler/Kjøkkenbord	kjøkkenbord
Møbler og interiør/Bord og stoler/Kontorstoler	kontorstol

Møbler og interiør/Senger og madrasser/Madrasser	madrass
Møbler og interiør/Senger og madrasser/Senger	seng
Møbler og interiør/Senger og madrasser/Rammemadrasser	rammemadrass
Møbler og interiør/Kjøkkenutstyr og serviser	kjøkkenutstyr,servise
Møbler og interiør/Belysning/Taklamper	taklampe
Møbler og interiør/Belysning/Bordlamper	bordlampe
Møbler og interiør/Belysning/Stålamper	stålampe
Møbler og interiør/Belysning/Vegglamper	vegglampe,lampett
Møbler og interiør/Hyller og kommoder/Hyller	hylle
Møbler og interiør/Hyller og kommoder/Skjenk	skjenk
Møbler og interiør/Hyller og kommoder/TV-møbler	tv-møbel
Møbler og interiør/Hyller og kommoder/Nattbord	nattbord
Møbler og interiør/Hyller og kommoder/Kommoder	kommode
Møbler og interiør/Garderobe og skap/Skap	skap
Møbler og interiør/Garderobe og skap/Garderober	garderobe
Møbler og interiør/Garderobe og skap/Seksjoner	seksjon
Møbler og interiør/Tepper og tekstiler/Duker	duker,duk
Møbler og interiør/Tepper og tekstiler/Gardiner	gardin,gardiner
Møbler og interiør/Tepper og tekstiler/Puter	puter,pute
Møbler og interiør/Tepper og tekstiler/Tepper	teppe
Antikviteter og kunst/Sølvtøy og bestikk	sølvtøy,bestikk
Utstyr til bil, båt og MC/Bildeler/Bilseter	bilsete
Utstyr til bil, båt og MC/Bildeler/Takstativ og boks	takstativ,takboks
Utstyr til bil, båt og MC/Bildeler/GPS	gps,navigasjonssystem
Utstyr til bil, båt og MC/Bildeler/Dekk og felger	dekk,felger
Utstyr til bil, båt og MC/Tilhengere/Hestehengere	hestehenger
Utstyr til bil, båt og MC/Tilhengere/Båt- og opplagshengere	båthenger,opplagshenger
Utstyr til bil, båt og MC/Tilhengere/Biltransporthengere	biltransporthenger
Utstyr til bil, båt og MC/Tilhengere/Tipphengere	tipphenger
Utstyr til bil, båt og MC/Tilhengere/Skaphengere	skaphenger
Utstyr til bil, båt og MC/Tilhengere/Varehengere	varehenger
Utstyr til bil, båt og MC/MC-deler/MC-klær	mc-dress
Hage, oppussing og hus/Byggevarer og oppussing/Byggesett	anneks,bod
Hage, oppussing og hus/Byggevarer og oppussing/Tapet og maling	tapet,maling
Hage, oppussing og hus/Byggevarer og oppussing/Vinduer	vind
Hage, oppussing og hus/Byggevarer og oppussing/Trapper	trapper
Hage, oppussing og hus/Byggevarer og oppussing/Gulv	gulv,gulvbelegg
Hage, oppussing og hus/Byggevarer og oppussing/Dører	dør
Hage, oppussing og hus/Garasje	garasje,garasjeport
Hage, oppussing og hus/Kjøkkeninnredning	kjøkkeninnredning
Hage, oppussing og hus/Hage og uteområder/Snørydding	snøfreser
Hage, oppussing og hus/Hage og uteområder/Grill	grill,gassgrill
Hage, oppussing og hus/Hage og uteområder/Hageredskap	motorsag,hekksaks
Hage, oppussing og hus/Hage og uteområder/Partytelt	partytelt
Hage, oppussing og hus/Hage og uteområder/Gressklippere	gressklipper
Hage, oppussing og hus/Oppvarming og ventilasjon/Peis	peis
Hage, oppussing og hus/Oppvarming og ventilasjon/Elektriske ovner	panelovn
Hage, oppussing og hus/Oppvarming og ventilasjon/Ved og brensel	fyringsved,brensel
Hage, oppussing og hus/Oppvarming og ventilasjon/Ovner	ovn
Hage, oppussing og hus/Baderomsinnredning/Dusjkabinett og -vegger	dusjkabinett
Hage, oppussing og hus/Baderomsinnredning/Servanter	servant,vaskeservant
Hage, oppussing og hus/Baderomsinnredning/Toaletter	toalett

Hage, oppussing og hus/Baderomsinnredning/Steamdusjer	steamdusj
Hage, oppussing og hus/Baderomsinnredning/Blandebatterier	blandebatteri
Hage, oppussing og hus/Baderomsinnredning/Badekar	badekar
Hage, oppussing og hus/Baderomsinnredning/Baderomsmøbler	baderomsmøbler
Sport og friluftsliv/Treningsapparater og -utstyr/Tredemøller	tredemølle
Sport og friluftsliv/Treningsapparater og -utstyr/Romaskiner	romaskin
Sport og friluftsliv/Treningsapparater og -utstyr/Ellipsemaskiner	ellipsemaskin
Sport og friluftsliv/Treningsapparater og -utstyr/Styrkeapparater	vekter
Sport og friluftsliv/Treningsapparater og -utstyr/Treningssykler	trimsykkel,ergometersykkel
Sport og friluftsliv/Treningsapparater og -utstyr/Stepmaskiner	stepmaskin
Sport og friluftsliv/Ballsport/Basketball	basketball
Sport og friluftsliv/Ballsport/Bandy	bandy
Sport og friluftsliv/Ballsport/Tennis	tennis,tennisracket
Sport og friluftsliv/Ballsport/Fotball	fotball
Sport og friluftsliv/Sportsskyting/Softgun	softgun
Sport og friluftsliv/Sportsskyting/Paintball	paintball
Sport og friluftsliv/Sportsskyting/Luftvåpen	luftpistol,luftgevær
Sport og friluftsliv/Vannsport/Dykking	dykking,dykkeutstyr
Sport og friluftsliv/Vannsport/Padling	padling
Sport og friluftsliv/Vannsport/Surfing	surfing,surfebrett
Sport og friluftsliv/Vannsport/Windsurf	windsurfing
Sport og friluftsliv/Skøytesport	skøyter
Sport og friluftsliv/Sykkelsport/Sykler/Fulldemper	fulldemper,fulldempet
Sport og friluftsliv/Sykkelsport/Sykler/Cyclocross	cyclocross
Sport og friluftsliv/Sykkelsport/Sykler/BMX	bmx
Sport og friluftsliv/Sykkelsport/Sykler/Hybrid/bysykkel	hybridsykkel,bysykkel
Sport og friluftsliv/Sykkelsport/Sykler/Elektriske	elsykkel,el-sykkel
Sport og friluftsliv/Sykkelsport/Sykler/Landevei	racersykkel,landeveissykkel
Sport og friluftsliv/Sykkelsport/Sykler/Terreng	mountainbike,terrengsykkel
Sport og friluftsliv/Skisport/Telemark	telemarkski
Sport og friluftsliv/Skisport/Snowboard	snowboard,snøbrett
Sport og friluftsliv/Skisport/Alpint	alpint
Sport og friluftsliv/Skisport/Langrenn	langrennsski
Sport og friluftsliv/Golf	golfkøller
Sport og friluftsliv/Ekstremsport/Skate- og longboard	skateboard,longboard
Sport og friluftsliv/GPS og pulsklokker	pulsklokke
Sport og friluftsliv/Jakt, fiske og friluftsliv/Sneller, fluer og kroker	snelle,fluer
Sport og friluftsliv/Jakt, fiske og friluftsliv/Fiskestenger	fiskestenger,fiskestang
Sport og friluftsliv/Jakt, fiske og friluftsliv/Våpen	jaktvåpen
Sport og friluftsliv/Jakt, fiske og friluftsliv/Kikkerter og optikk	kikkert,kikkerter
Sport og friluftsliv/Jakt, fiske og friluftsliv/Ryggsekker	ryggsekk,tursekk
Sport og friluftsliv/Jakt, fiske og friluftsliv/Kano	kano
Sport og friluftsliv/Jakt, fiske og friluftsliv/Telt	telt
Sport og friluftsliv/Jakt, fiske og friluftsliv/Soveposer	sovepose
Klær, kosmetikk og accessoirer/Vesker og bager	veske,bag
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Ørepynt	ørepynt
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Armbånd	armbånd
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Ringer	gullring,diamantring
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Halskjeder	halskjede
Klær, kosmetikk og accessoirer/Smykker og oppbevaring/Smykkeskrin og oppbevaring	smykkeskrin
Klær, kosmetikk og accessoirer/Herreklær/T-skjorter	t-skjorte
Klær, kosmetikk og accessoirer/Herreklær/Bukser	bukse

Klær, kosmetikk og accessoirer/Herreklær/Luer,skjerf og votter	lue,skjerf,votter
Klær, kosmetikk og accessoirer/Herreklær/Undertøy	undertøy
Klær, kosmetikk og accessoirer/Herreklær/Smokinger	smoking
Klær, kosmetikk og accessoirer/Herreklær/Gensere	herregenser
Klær, kosmetikk og accessoirer/Herreklær/Ytterklær	ytterklær,ytterjakke
Klær, kosmetikk og accessoirer/Herreklær/Dresser	dress
Klær, kosmetikk og accessoirer/Herreklær/Skjorter	herreskjorte
Klær, kosmetikk og accessoirer/Herreklær/Bunader	herrebunad
Klær, kosmetikk og accessoirer/Sko/Damesko	damesko
Klær, kosmetikk og accessoirer/Sko/Herresko	herresko
Klær, kosmetikk og accessoirer/Dameklær/T-skjorter	t-skjorte
Klær, kosmetikk og accessoirer/Dameklær/Bukser	damebukse
Klær, kosmetikk og accessoirer/Dameklær/Jakker	damejakke
Klær, kosmetikk og accessoirer/Dameklær/Luer,skjerf og votter	lue,skjerf,votter
Klær, kosmetikk og accessoirer/Dameklær/Brudekjoler	brudekjole
Klær, kosmetikk og accessoirer/Dameklær/Undertøy	dameundertøy
Klær, kosmetikk og accessoirer/Dameklær/Kjoler	kjole
Klær, kosmetikk og accessoirer/Dameklær/Topper	topper
Klær, kosmetikk og accessoirer/Dameklær/Gensere	genser
Klær, kosmetikk og accessoirer/Dameklær/Ytterklær	ytterklær,ytterjakke
Klær, kosmetikk og accessoirer/Dameklær/Skjorter	bluse
Klær, kosmetikk og accessoirer/Dameklær/Skjørt	skjørt
Klær, kosmetikk og accessoirer/Dameklær/Bunader	beltestakk
Klær, kosmetikk og accessoirer/Klokker og ur	klokke
Klær, kosmetikk og accessoirer/Briller og linser/Solbriller	solbriller
Klær, kosmetikk og accessoirer/Briller og linser/Briller	briller
Klær, kosmetikk og accessoirer/Kosmetikk	kosmetikk
Foreldre og barn/Barneklær og sko/Gutt	gutteklær
Foreldre og barn/Barneklær og sko/Jente	jenteklær
Foreldre og barn/Barneklær og sko/Baby	babyklær
Foreldre og barn/Barneseter	barnesete
Foreldre og barn/Barnevogner	barnevogn
Foreldre og barn/Barnemøbler/Stoler	barnestol
Foreldre og barn/Barnemøbler/Stellebord	stellebord
Foreldre og barn/Barnemøbler/Senger	barneseng

7.3 Removed stopwords

Removed stopwords : og, også, som, eller, å, ellers, men, så, med, på, til, fra, av, hos, ut, bak, blant, etter, for, gjennom, i, ifølge, innen, innenfor, mellom, mot, om, omkring, ovenfor, rundt, under, ved, kan, må, skal, vil, den, en, et, enn, det, denne, dette, disse, da, er, var, har, blir, hadde, vært, ble, jeg, deg, du, dere, han,hun, vi, de, dem, seg, man

References

Özlem Aslan, Xinhua Zhang, and Dale Schuurmans. Convex deep learning via normalized kernels. In *Advances in Neural Information Processing Systems*, pages 3275–3283, 2014.

Aurélien Bellet, Amaury Habrard, and Marc Sebban. A survey on metric learning for feature vectors and structured data. *arXiv preprint arXiv:1306.6709*, 2013.

David M Blei, Andrew Y Ng, and Michael I Jordan. Latent dirichlet allocation. *the Journal of machine Learning research*, 3:993–1022, 2003.

Steve Branson, Grant Van Horn, Serge Belongie, and Pietro Perona. Bird species categorization using pose normalized deep convolutional nets. *arXiv preprint arXiv:1406.2952*, 2014.

Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

Rakesh Chalasani and Jose C Principe. Deep predictive coding networks. *arXiv preprint arXiv:1301.3541*, 2013.

Tsung-Han Chan, Kui Jia, Shenghua Gao, Jiwen Lu, Zinan Zeng, and Yi Ma. Pcanet: A simple deep learning baseline for image classification? *arXiv preprint arXiv:1404.3606*, 2014.

Moses S Charikar. Similarity estimation techniques from rounding algorithms. In *Proceedings of the thiry-fourth annual ACM symposium on Theory of computing*, pages 380–388. ACM, 2002.

Gal Chechik, Varun Sharma, Uri Shalit, and Samy Bengio. Large scale online learning of image similarity through ranking. *The Journal of Machine Learning Research*, 11:1109–1135, 2010.

Youngmin Cho and Lawrence K Saul. Kernel methods for deep learning. In *Advances in neural information processing systems*, pages 342–350, 2009.

Anna Choromanska, Mikael Henaff, Michael Mathieu, Gérard Ben Arous, and Yann LeCun. The loss surface of multilayer networks. *arXiv preprint arXiv:1412.0233*, 2014.

George E Dahl, Dong Yu, Li Deng, and Alex Acero. Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition. *Audio, Speech, and Language Processing, IEEE Transactions on*, 20(1):30–42, 2012.

Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 886–893. IEEE, 2005.

Scott C. Deerwester, Susan T Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JAsIs*, 41(6): 391–407, 1990.

Kunihiko Fukushima. Cognitron: A self-organizing multilayered neural network. *Biological cybernetics*, 20(3-4):121–136, 1975.

Yarin Gal and Zoubin Ghahramani. Dropout as a bayesian approximation: Insights and applications. In *Deep Learning Workshop, ICML*, 2015.

Xavier Glorot and Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. In *International conference on artificial intelligence and statistics*, pages 249–256, 2010.

Ian J Goodfellow and Oriol Vinyals. Qualitatively characterizing neural network optimization problems. *arXiv preprint arXiv:1412.6544*, 2014.

Kristen Grauman and Rob Fergus. Learning binary hash codes for large-scale image search. In *Machine learning for computer vision*, pages 49–87. Springer, 2013.

Thomas L Griffiths and Mark Steyvers. Finding scientific topics. *Proceedings of the National Academy of Sciences*, 101(suppl 1):5228–5235, 2004.

Tom Griffiths. Gibbs sampling in the generative model of latent dirichlet allocation. 2002.

Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *International Conference on Artificial Intelligence and Statistics*, pages 297–304, 2010.

LI Hang. A short introduction to learning to rank. *IEICE TRANSACTIONS on Information and Systems*, 94(10):1854–1862, 2011.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. *arXiv preprint arXiv:1502.01852*, 2015.

Geoffrey E Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, and Ruslan R Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

Sepp Hochreiter. Untersuchungen zu dynamischen neuronalen netzen. *Diploma, Technische Universität München*, 1991.

Andrej Karpathy and Li Fei-Fei. Deep visual-semantic alignments for generating image descriptions. *arXiv preprint arXiv:1412.2306*, 2014.

Douwe Kiela and Léon Bottou. Learning image embeddings using convolutional neural networks for improved multi-modal semantics. In *Proceedings of EMNLP*, volume 2014, 2014.

Ryan Kiros, Ruslan Salakhutdinov, and Richard S Zemel. Unifying visual-semantic embeddings with multimodal neural language models. *arXiv preprint arXiv:1411.2539*, 2014.

Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

Brian Kulis. Metric learning: A survey. *Foundations and Trends in Machine Learning*, 5(4):287–364, 2012.

Quoc V Le and Tomas Mikolov. Distributed representations of sentences and documents. *arXiv preprint arXiv:1405.4053*, 2014.

Yann LeCun, Bernhard Boser, John S Denker, Donnie Henderson, Richard E Howard, Wayne Hubbard, and Lawrence D Jackel. Backpropagation applied to handwritten zip code recognition. *Neural computation*, 1(4):541–551, 1989.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems*, pages 2177–2185, 2014.

Omer Levy, Yoav Goldberg, and Ido Dagan. Improving distributional similarity with lessons learned from word embeddings. *Transactions of the Association for Computational Linguistics*, 3:211–225, 2015.

M. Lin, Q. Chen, and S. Yan. Network in network. *arXiv preprint arXiv:1312.4400*, 2013.

David G Lowe. Object recognition from local scale-invariant features. In *Computer vision, 1999. The proceedings of the seventh IEEE international conference on*, volume 2, pages 1150–1157. Ieee, 1999.

Tomasz Malisiewicz. From feature descriptors to deep learning: 20 years of computer vision, 2015. URL http://www.computervisionblog.com/2015/01/from-feature-descriptors-to-deep.html.

Pankaj Mehta and David J Schwab. An exact mapping between the variational renormalization group and deep learning. *arXiv preprint arXiv:1410.3831*, 2014.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119, 2013.

Thomas Minka. Estimating a dirichlet distribution, 2000.

Thomas Minka and John Lafferty. Expectation-propagation for the generative aspect model. In *Proceedings of the Eighteenth conference on Uncertainty in artificial intelligence*, pages 352–359. Morgan Kaufmann Publishers Inc., 2002.

Guido F Montufar, Razvan Pascanu, Kyunghyun Cho, and Yoshua Bengio. On the number of linear regions of deep neural networks. In *Advances in Neural Information Processing Systems*, pages 2924–2932, 2014.

Mohammad Norouzi, Tomas Mikolov, Samy Bengio, Yoram Singer, Jonathon Shlens, Andrea Frome, Greg S Corrado, and Jeffrey Dean. Zero-shot learning by convex combination of semantic embeddings. *arXiv preprint arXiv:1312.5650*, 2013.

Frank Rosenblatt. The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological review*, 65(6):386, 1958.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning representations by back-propagating errors. *Cognitive modeling*, 5:3, 1988.

Richard Socher, Cliff C Lin, Chris Manning, and Andrew Y Ng. Parsing natural scenes and natural language with recursive neural networks. In *Proceedings of the 28th international conference on machine learning (ICML-11)*, pages 129–136, 2011.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958, 2014.

Ilya Sutskever, James Martens, George Dahl, and Geoffrey Hinton. On the importance of initialization and momentum in deep learning. In *Proceedings of the 30th international conference on machine learning (ICML-13)*, pages 1139–1147, 2013.

Aaron Van den Oord, Sander Dieleman, and Benjamin Schrauwen. Deep content-based music recommendation. In *Advances in Neural Information Processing Systems*, pages 2643–2651, 2013.

Laurens Van der Maaten and Geoffrey Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(2579-2605):85, 2008.

Oriol Vinyals, Alexander Toshev, Samy Bengio, and Dumitru Erhan. Show and tell: A neural image caption generator. *arXiv preprint arXiv:1411.4555*, 2014.

Hanna M Wallach, David M Mimno, and Andrew McCallum. Rethinking lda: Why priors matter. In *Advances in neural information processing systems*, pages 1973–1981, 2009.

Ji Wan, Dayong Wang, Steven Chu Hong Hoi, Pengcheng Wu, Jianke Zhu, Yongdong Zhang, and Jintao Li. Deep learning for content-based image retrieval: A comprehensive study. In *Proceedings of the ACM International Conference on Multimedia*, pages 157–166. ACM, 2014.

Jiang Wang, Yang Song, Tommy Leung, Catherine Rosenberg, Jingbin Wang, James Philbin, Bo Chen, and Ying Wu. Learning fine-grained image similarity with deep ranking. In *Computer Vision and Pattern Recognition (CVPR), 2014 IEEE Conference on*, pages 1386–1393. IEEE, 2014.

Jason Yosinski, Jeff Clune, Yoshua Bengio, and Hod Lipson. How transferable are features in deep neural networks? In *Advances in Neural Information Processing Systems*, pages 3320–3328, 2014.

Matthew D Zeiler and Rob Fergus. Visualizing and understanding convolutional networks. In *Computer Vision–ECCV 2014*, pages 818–833. Springer, 2014.